Secrecy-Oriented First-Order Logical Analysis of Cryptographic Protocols

Gergei Bana
SQIG - Instituto de Telecomunicações and
Department of Mathematics
IST, Technical University of Lisbon, Portugal
bana@math.upenn.edu

Koji Hasebe
Graduate School of Systems and
Information Engineering
University of Tsukuba, Japan
hasebe@iit.tsukuba.ac.jp

Mitsuhiro Okada
Department of Philosophy
Keio University, Tokyo, Japan
mitsu@abelard.flet.keio.ac.jp

Abstract

We present a computationally sound first-order system for security analysis of protocols that places secrecy of nonces and keys in its center. Even trace properties such as agreement and authentication are proven via proving a non-trace property, namely, secrecy first. This results a very powerful system, the working of which we illustrate on the agreement and authenti- cation proofs for the Needham-Schroeder-Lowe public-key and the amended Needham-Schroeder shared-key protocols in case of unlimited sessions. Unlike other available formal verification techniques, computational soundness of our approach does not require any idealizations about parsing of bitstrings or unnecessary tagging. In particular, we have total control over detecting or eliminating the possibility of type-flaw attacks.

Keywords. *cryptographic protocols, formal methods, first order logic, computational semantics*

1. Introduction

In the first-order logic framework, protocol correctness is analyzed by defining a syntax with adding some additional axioms (expressing security properties etc.) to the usual axioms and inference rules of first order logic and then proving some security property directly. This is in contrast with other methods that eliminating the possibility of successful formal (Dolev-Yao) adversaries. In those other approaches, for computational soundness, it is necessary to show that the lack of successful formal adversaries lead to a lack of successful computational ones. However, no-one has managed to do this satisfactorily, as such a theorem always requires to limit the capabilities of computational adversaries as there is no complete formal description of computational adversarial capabilities. In contrast, a logical method, like the one we present, does not require the notion of formal adversary. The link to the computational world is done by assigning a class of computational structures to the syntax, proving that the axioms and inference rules hold (computationally) there, and so the elements of this class can work as computational models of the syntax. This way, a property provable in the syntax must be true in any such computational model.

In earlier logical methods as [13, 10, 2], trace properties were not proven via secrecy of encryption, but, instead, via rather limited aspects of the secrecy of encryption. Namely, in case of public key, via axioms saying that if something was encrypted honestly with an honest agent's public key, and this thing later appeared in a different form, then it had to go through the agent whose key was used for the encryption. Hence, the strong notion of computational secrecy was limited to this trace aspect of it. Instead, we now consider the following predicates ($Q\ acts_i t'$ is a send or receive action of Q in session i) $Sec(\langle A_1, ..., A_n\rangle, t, N); Q\ acts_i\ t'$ and $KeySec(\langle A_1, ..., A_n\rangle, t, K); Q\ acts_i\ t'$ The meaning of the first is the following: agents not included in $A_1, ..., A_n$, and also the adversary, based on their views of the protocol until (not included) the action $Q\ acts_i\ t'$, and providing them t, they cannot differentiate the nonce N from another nonce N' that was generated independently of the protocol. The meaning of the second one is that the key K can be used for secure encryption by any of $A_1, ..., A_n$ even if t is revealed to the public.

The rough idea of protocol proof is then to derive from a set of axioms that if $A_1, ..., A_n$ are honest participants of the protocol carrying out only their protocol roles, then for each of their send ac-

tions, $[Key]Sec(\langle A_1, ..., A_n\rangle, N/K); A_j \ sends_i \ t$ implies $[Key]Sec(\langle A_1, ..., A_n\rangle, t, N/K); A_j \ sends_i \ t$. Note that t in the send action and inside the secrecy predicate are the same. This means that if N (or K) was uncorrupted until sending t, then the send action will not corrupt N (or K), as the secrecy of it holds even if t is revealed. The core part of the protocol proof is to build t up inside the security predicate from its parts (e.g. $t = \{N_1, A\}^r_A$). This is proven for every send action of every honest agent. Hence, this means that N and K are never corrupted, because no send action corrupts them. For the commitment problem, the formulation is a little different, proving that N or K are not corrupted *before a certain action happens*.

Once this way the uncorrupted nature of nonces and keys are proven, the agreement and authentication properties can be proven, as when uncorrupted nonces are received encrypted, they must have been sent by agents with respect of whom the nonce is secret.

We also present a theorem, which says that for proving a certain class of formulas (agreement and authentication are such) the proof can be carried out in a simpler theory, which is not sound though.

Recently, Roy et. al. [19, 18] created a system for proving secrecy inductively in PCL. However, they define intermediate trace properties, which we don't need, but use induction directly on non-trace properties. Our system is rather simple: it is a standard first order theory, we did not need the use of Hoare logic. Further, if we understand correctly, they do not prove authentication via secrecy, and we do not need elaborate methods to avoid key cycles either. Moreover, earlier in [2], we showed a number of problems with their computational semantics, avoidable only with tagging. In our system, no tagging is necessary. Further, a problem of PCL is that none of their computational papers clarify the full set of axioms used in their proofs, no sound term axioms have been provided. In this paper we show the full set of our axioms that we use.

This system hence has a number of advantages:

- It is a relatively simple, but very general system that can provide computationally sound proofs of secrecy, authentication, agreement for unlimited sessions.
- Does not require any idealizations about parsing bitstrings or tagging for computational soundness.
- Parsing properties can be conveniently adjusted in the term axioms. As a result, type-flaw errors in the form of missing axioms for a proof can be detected, and the set of axioms can be modified depending on what kind of type-flaw errors we want to allow.
- As far as we know, for the first time without idealized parsing, provides computationally sound agreement, authentication and secrecy proofs for NSL and symmetric NS protocols (with IND-CCA public and IND-CCA as well as for given plaintext unforgeable public key encryptions).
- As far as we know, for the first time able to deal with the commitment problem.
- The system implicitly also takes care of key cycles. The axioms are such, that in case of key cycles, it is not possible to prove $KeySec$ for those keys, hence encryptions with them cannot be used for proving secrecy or authentication.

Related Work. Formal methods emerged from the seminal work of Dolev and Yao [12]. The main approaches have been on one hand logical such as in BAN logic [4], Protocol Composition Logic (PCL) [13], on the other hand trace-based, such as strand spaces [14] or MSR [6]. Accordingly, linking formal and computational security in an active setting has two major different approaches: a logical with no formal execution as in [11, 2], or deriving computational security from the non-existence of formal attacks [1, 9, 8]. Recently Roy et al. considered inductive derivation of security in computational setting in [18].

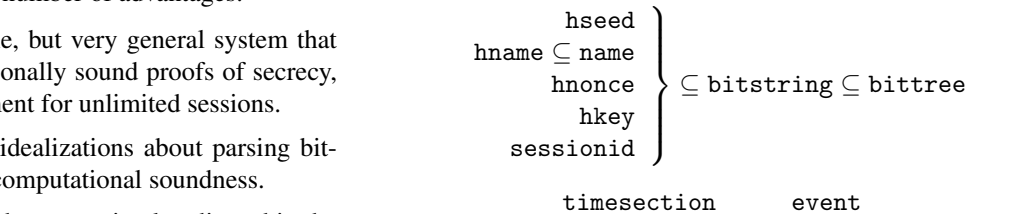
We would like to thank Paulo Mateus and Hubert Comon-Lundh for many inspiring discussions.

2. Basic Protocol Logic 2.0 Syntax

For the intuitive meaning of syntactic objects, let's clarify the following: A computational execution of a protocol is stochastic, with a sequence of coin tosses defining a *tree of coin tosses* for each fixed security parameter. We will call one branch, when all coins are fixed, a *branch of coin tosses*. The set of all branches is the underlying probability space.

2.1. Language

Sorts. Our language is order-sorted, with the following sort structure:



We require countably infinite variables of each sort. `timesection` has three constant, $\mathbf{0}$, $\mathbf{1}$ and ∞, while `event` has one: $\mathbf{D}$. We also introduce the error constant $\perp$ of sort `bitstring`.

The intuitive meanings of the sorts are the following:

- `hname` is the sort of the names of honest (uncorrupted) principals (we don't model dynamic corruption here).

They as well as their long-term keys have to be fixed before the run of the protocol.

- `name` represent principals in general. They don't have to be fixed before the run of the protocol. Their keys do not have to be correctly generated. Any principal's name may have non-trivial distributions. They are known to all principals and the adversary.
- `hnonce` means honest nonces, which are honestly generated by some principal, with the correct distribution and independently of everything that happened before.
- `hkey` means honest shared keys, which are honestly generated by some principal with the correct distribution, independently of everything happening before.
- `hseed` represent honestly generated random seed of encryptions.
- `sessionid` represents the session id's that keep track of the principals' sessions. These are internal records.
- `bitstring` represents unparsed messages. Computationally, a sequence (in the security parameter) of bitstring valued random variables (functions) over a non-negligible part of the tree of coin tosses.
- `bittree` represents parsed messages. That is, labeled ordered trees such that the leafs are labelled by items of sort `bitstring`, while the internal nodes are labelled by one of the function symbols $Pair$, $LPKEnctypt$, $LSKEnctypt$, $SKEnctypt$ (see below). The child nodes have to agree with the arities of the function symbols.
- `timesection` represents a moment of the execution. Computationally, it assigns a natural number to every branch of coin tosses, so it is a sequence of random variables.
- `event` represents probabilistic events. Computationally, non-negligible sequences (in the security parameter) of sets of branches of coin tosses.

The computational interpretations of the sorts in Section A.2 should clarify their meaning further.

Remark 2.1. The sort `event` is only needed for soundness. In the authentication and agreement proofs, the proofs will be correct even if we delete this sort from the axioms. However, in that case, the axioms will not be sound any more. See Theorem 2.10.

We will denote variables according the Table 1.

Function Symbols and Terms. We introduce the following function symbols with short notation. Please note from now that the intuitive meaning of terms here is different from what is common in computational semantics. $\{t\}^s_Q$ is not a ciphertext, but a parsed object. $\overline{\{t\}^s_Q}$ is the ciphertext.

`hseed`	$r, r', \ldots, r_1, r_2$
`hname`	$A, B, \ldots, A_1, A_2, \ldots$
`name`	$Q, Q' \ldots, Q_1, Q_2, \ldots$
`hnonce`	$N, N', \ldots, N_1, N_2, \ldots$
`hkey`	$K, K', \ldots, K_1, K_2, \ldots$
`nonce` or `hkey`	$\nu, \nu', \ldots, \nu_1, \nu_2, \ldots$
`sessionid`	$i, i', \ldots, i_1, i_2, \ldots$
`bitstring`	$s, s', \ldots, s_1, s_2, \ldots$ $n, n', \ldots, n_1, n_2, \ldots$ $k, k', \ldots, k_1, k_2, \ldots$ $m, m', \ldots, m_1, m_2, \ldots$
`bittree`	$t, t', \ldots, t_1, t_2, \ldots$ $u, u', \ldots, u_1, u_2, \ldots$
`timesection`	$\tau, \tau', \ldots, \tau_1, \tau_2, \ldots$
`event`	$\Delta, \Delta', \ldots, \Delta_1, \Delta_2, \ldots$
any variable	$v, v', \ldots, v_1, v_2, \ldots$

Table 1. Notation of variables

- Pairing:

$$Pair : \texttt{bittree} \times \texttt{bittree} \to \texttt{bittree}$$
$$\langle t, t' \rangle \equiv Pair(t, t')$$
$$\langle t_1, t_2, t_3 ..., t_j \rangle \equiv \langle ...\langle\langle t_1, t_2\rangle, t_3\rangle, ..., t_j\rangle$$

- Encryption with (long term) public key (of Q):

$$LPKEnctypt : \texttt{name} \times \texttt{bittree} \times \texttt{bitstring} \to \texttt{bittree}$$
$$\{t\}^s_Q \equiv LPKEncrypt(Q, t, s)$$

- Encryption with long term shared key (of Q and Q'):

$$LSKEnc : \texttt{name} \times \texttt{name} \times \texttt{bittree} \times \texttt{bitstring} \to \texttt{bittree}$$
$$\{t\}^s_{QQ'} \equiv LSKEncrypt(Q, Q', t, s)$$

- Encryption with shared session key:

$$SKEnc : \texttt{bitstring} \times \texttt{bittree} \times \texttt{bitstring} \to \texttt{bittree}$$
$$\{t\}^s_k \equiv SKEnc(k, t, s)$$

- Computation of the encryptions and pairings in a `bittree`:

$$Evaluate : \texttt{bittree} \to \texttt{bitstring}$$
$$\overline{t} \equiv Evaluate(t)$$

Message terms (we refer to them as "terms" in short, although strictly speaking τ and Δ are also terms as usual in first order logic) are defined as:

$$T ::= t \;\Big|\; \overline{T} \;\Big|\; \langle T, T\rangle \;\Big|\; \{T\}^s_Q \;\Big|\; \{T\}^s_{QQ'} \;\Big|\; \{T\}^s_k.$$

All message terms are of sort `bittree`. We will sometimes use the meta-symbols $T, T' ...$ to denote compound terms. We think of a term as the following tree. The expression under the overline is not parsed as it is a `bitstring`.

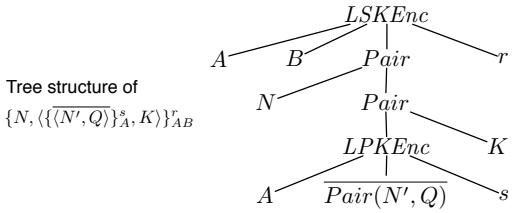


Figure 1. Parsing tree

Predicates. We introduce a number of predicate symbols, the intuitive meanings of which require some explanation.

- $t = t'$, $\tau = \tau'$, $\Delta = \Delta'$
  Equality up to negligible probability. For t and t' trees, the must have the same tree structure with the same function symbol labels, and equal labels on the leafs.
- $t = t'\big|_\Delta$, $\tau = \tau'\big|_\Delta$, $\Delta' = \Delta''\big|_\Delta$
  Ternary predicates. Equality up to negligibility restricted to the event (subset) Δ. We will also use the notation $t =_\Delta t'$, $\tau =_\Delta \tau'$, $\Delta' =_\Delta \Delta''$. Predicate $=_{\mathbf{D}}$ is the same as $=$, so $=$ is a special case of $=_{\mathbf{D}}$.
- $t \sqsubseteq t'\big|_\Delta$
  Ternary predicate. Meaning that the tree of t equals on Δ (in the above sense) a subtree of t', and is not a leaf corresponding to the random feed or the key of an encrypion. We will also use $\sqsubseteq_\Delta$. We also wright $\sqsubseteq$ for $\sqsubseteq_{\mathbf{D}}$. See examples below.
- $\tau < \tau'\big|_\Delta$
  Ternary. Time section τ is strictly earlier than τ' of every branch of Δ (up to negligible probability).
- $\Delta' \subseteq \Delta''\big|_\Delta$
  The $\Delta' \cap \Delta$ is a subset of $\Delta'' \cap \Delta$ (up to negligible probability).
- $Q\ generates^i_\tau N\big|_\Delta$
  Principal Q generates a fresh nonce N at time section τ in session he labels i on all (up to negligible probability) branches in Δ. Such N will be required to have the correct distribution of nonces in Δ and be independent of what happened until the generation.
- $Q\ receives^i_\tau t\big|_\Delta$
  On all branches of Δ, principal Q receives a message in session i at time τ represented by t and parses it to reveal the subterms in t. Q does exactly as much parsing as shown by t (except for randomnesses in the upper index of encryption which is not parsed). That is, $Q\ receives^i_\tau \{A, m, \overline{\langle N, K\rangle}\}^r_Q$ means that Q received the message that as a random variable of bitstrings is $\overline{\{A, m, \overline{\langle N, K\rangle}\}^r_Q}$, and parsed it to the point of recording A and m and the bit string $\overline{\langle N, K\rangle}$ (again, r is never parsed, and $\overline{\langle N, K\rangle}$ is not parsed either as it is not a tree, but a bit string). Note, if Q is honest, it only parses using its own decryption key, so $Q\ receives^i_\tau \{A, M\}^r_{Q'}$ implies $Q = Q'$.
- $Q\ sends^i_\tau t\big|_\Delta$
  The meaning of this is similar to the receive action, but in the opposite direction: it means on all branches of Δ, Q in its session i puts together t from as many parts as indicated in the expression, computes $\bar{t}$ and then sends it at time τ.
- $Sec_\tau(\langle A_1, ...A_n\rangle, t, \nu)\big|_\Delta$
  This is called *secrecy predicate*. It is satisfied if principals other $A_1, ...A_n$ together with the adversary cannot distinguish ν from a nonce (or key respectively) generated independently of the protocol based on their view of the protocol until (not including) τ in Δ, even if they are given the bitstring corresponding to t, that is, $\bar{t}$.
- $KeySec_\tau(\langle A_1, ...A_n\rangle, t, K)\big|_\Delta$
  This is called *key-secrecy predicate*. It is satisfied if $A_1, ...A_n$ can safely encrypt messages using K until (not including) τ in Δ, even if t is given to them.

If a predicate does not contain $|_\Delta$, it is meant to be $|_{\mathbf{D}}$.

We will write $[Key]Sec$ when we mean Sec or $KeySec$. In case of multiple occurrence of $[Key]Sec$ in a sentence or formula, then either all are Sec or all are $KeySec$.

Since we do not have a special sort for lists of names, we extend $[Key]Sec$ to $[Key]Sec(u, t, \nu)$ so that if u is not a list of honest names, then the predicate gives false. We will use the notation $\vec{A} \equiv \langle A_1, ...A_n\rangle$.

We write $[Key]Sec_\tau(\vec{A}, \nu)$ when no additional t is revealed. This is the same as $[Key]Sec_\tau(\vec{A}, A, \nu)$ for any A, as A is known by everyone, so it does not reveal any additional information about ν.

Example 2.2. In order to make the term structure more understandable, we include a few examples here for equality and for subterm relation: if $N \neq Q$, then $Q \not\sqsubseteq \{N\}^s_{QQ'}$; if $N \neq s$, then $s \not\sqsubseteq \{N\}^s_{QQ'}$; if $N \neq N'$, then $N \not\sqsubseteq \{\overline{\{N\}^r_B}, N'\}^s_{QQ}$; $\overline{\{N\}^r_B} \sqsubseteq \{\overline{\{N\}^r_B}, N'\}^s_{QQ}$. Notice, that terms under the overline are not considered subterms. Variables in the indices are not subterms either. The terms $\{N\}^s_{QQ'}$ and $\{N\}^{s'}_{QQ'}$ with different random seeds may, or may not be equal. They are equal if and only if they compute to the same bit string, that is, iff $\overline{\{N\}^s_{QQ'}} = \overline{\{N\}^{s'}_{QQ'}}$.

Formulas. Atomic formulas φ_0 are either of the forms

$$\varphi_0 ::= Q\ acts^i_\tau t\big|_\Delta \;\Big|\; [Key]Sec_\tau(u, t, \nu)\big|_\Delta$$
$$t =_\Delta t' \;\Big|\; t \sqsubseteq_\Delta t' \;\Big|\; \tau <_\Delta \tau' \;\Big|\; \Delta' \subseteq_\Delta \Delta''$$

Compound formulas are defined by

$$\varphi ::= \varphi_0 \Big| \neg\varphi \Big| \varphi \wedge \varphi \Big| \varphi \vee \varphi \Big| \varphi \to \varphi \Big| \forall v\varphi \Big| \exists v\varphi$$

where v is any variable.

We use the meta expression $\varphi[\vec{v}]$ to indicate the list of all free variables $\vec{v}$ occurring in φ. Let $\varphi|_\Delta$ be an abbreviation defined as follows. Let φ_0 an atomic expression no $|_\Delta$ restriction.

- $(\varphi_0|_{\Delta'})|_\Delta \equiv (\Delta' \subseteq \Delta \rightarrow \varphi_0|_{\Delta'}) \wedge (\Delta \subseteq \Delta' \rightarrow \varphi_0|_\Delta)$
- $(\varphi_1 \wedge \varphi_2)|_\Delta \equiv \varphi_1|_\Delta \wedge \varphi_2|_\Delta$,
- $(\varphi_1 \vee \varphi_2)|_\Delta \equiv \varphi_1|_\Delta \vee \varphi_2|_\Delta$
- $(\neg\varphi)|_\Delta \equiv \neg(\varphi|_\Delta)$
- $(\exists\Delta'\varphi)|_\Delta \equiv \exists\Delta'(\Delta' \subseteq \Delta \wedge \varphi|_\Delta)$
- $(\forall\Delta'\varphi)|_\Delta \equiv \forall\Delta'(\Delta' \subseteq \Delta \wedge \varphi|_\Delta)$

Let us introduce the notation:

$$Q_1\ acts_1^{i_1} t_1; ...; Q_k\ acts_k^{i_k} t_k \equiv$$
$$\exists\tau_1...\tau_k(\mathbf{0} < \tau_1 < ... < \tau_k \wedge Q_1\ acts_{1,\tau_1}^{i_1} t_1 \wedge ... \wedge Q_k\ acts_{\tau_k}^{i_k} t_k)$$

and

$$Q_1\ acts_{1,\tau_1}^{i_1} t_1; ...; Q_k\ acts_{k,\tau_k}^{i_k} t_k \equiv$$
$$\mathbf{0} < \tau_1 < ... < \tau_k \wedge Q_1\ acts_{1,\tau_1}^{i_1} t_1 \wedge ... \wedge Q_k\ acts_{\tau_k}^{i_k} t_k$$

These are called *trace formula*. Each $acts_j$ is one of *generates*, *sends* or *receives*. We also use $\alpha_1; ...; \alpha_k$ (or $\vec{\alpha}$ in short) to denote the formula $Q_1\ acts_1^{i_1} t_1; ...; Q_k\ acts_k^{i_k} t_k$. For $\vec{\alpha}$ ($\equiv \alpha_1; ...; \alpha_m$) and $\vec{\beta}$ ($\equiv \beta_1; ...; \beta_n$), we say $\vec{\beta}$ *includes* $\vec{\alpha}$, denoted by

$$\vec{\alpha} \subseteq \vec{\beta},$$

if there is a one-to-one, increasing function $j : \{1, ..., m\} \rightarrow \{1, ..., n\}$ such that $\alpha_k \equiv \beta_{j(k)}$.

Roles and Protocols. Roles of principals are described by trace formulas of the form

$$\vec{\alpha}^{A,i} \equiv A\ acts_1^i\ t_1; ...; A\ acts_k^i\ t_k,$$

where t_j's are not allowed to contain the *Evaluate* and *Restrict* function symbols. Protocols are a set of roles together with a list of values that the principals have to agree on.

Example 2.3. Roles of the Needham-Schroeder-Lowe protocol. We consider the Needham-Schroeder-Lowe public key protocol [16], whose informal description is as follows.

1. $A \rightarrow B$: $\{N_1, A\}_B$
2. $B \rightarrow A$: $\{N_1, N_2, B\}_A$
3. $A \rightarrow B$: $\{N_2\}_B$

Initiator's and responder's roles of the Needham-Schroeder-Lowe public key protocol (denoted by $Init_{\mathrm{NSL}}$ and $Resp_{\mathrm{NSL}}$, respectively) with session id's are described as the following formulas.

$Init_{\mathrm{NSL}}^A[A, i, Q, N_1, n_2, r_1, s_2, r_3] \equiv A\ generates^i\ N_1;$
$A\ sends^i\{N_1, A\}_Q^{r_1}; A\ receives^i\{N_1, n_2, Q\}_A^{s_2}; A\ sends^i\{n_2\}_Q^{r_3}$

$Resp_{\mathrm{NSL}}^B[B, i', Q', n_1, N_2, s_1, r_2, s_3] \equiv B\ receives^{i'}\{n_1, Q'\}_B^{s_1};$
$B\ generates^{i'} N_2; B\ sends^{i'}\{n_1, N_2, B\}_{Q'}^{r_2}; B\ receives^{i'}\{N_2\}_B^{s_3}$

They further have to agree that $Q = A$, $Q' = B$, $n_1 = N_1$, $n_2 = N_2$.

Remark 2.4. Notice that, for example, in the responder's role, we wrote $B\ receives^{i'}\{n_1, Q'\}_B^{s_1}$. We used n_1, Q', s_1, which, according to our notation are of sort `bitstring` because they may have been created by adversary and therefore, be arbitrarily distributed.

Example 2.5. Roles of the amended Needham-Schroeder shared-key protocol. We consider the amended Needham-Schroeder shared-key protocol [17, 7], whose informal description is as follows.

1. $A \rightarrow B : A$
2. $B \rightarrow A : \{A, N_1\}_{BT}$
3. $A \rightarrow T : \langle A, B, N_2, \{A, N_1\}_{BT}\rangle$
4. $T \rightarrow A : \{N_2, B, K, \{K, N_1, A\}_{BT}\}_{AT}$
5. $A \rightarrow B : \{K, N_1, A\}_{BT}$
6. $B \rightarrow A : \{N_3\}_K$
7. $A \rightarrow B : \{N_3, A\}_K$

In the original protocol, there is $N_3 - 1$ instead of $\langle N_3, A\rangle$, but for now we don't want to extend our syntax with additional functions. We have to assume that A and the pairing are such that $\langle N_3, A\rangle \neq N_3$.

The Initiator's, the responder's and the Trusted party's roles are described as the following formulas. We consider a constant T of sort `hname` for the trusted party as we don't want to quantify on T.

$Init_{sNS}^A[T, A, i, Q_2, m_1, m_2, k, N_2, n_3, s_3, s_4, r_5] \equiv$
$A\ sends^i A; A\ receives^i m_1; A\ generates^i N_2;$
$A\ sends^i\langle A, Q_2, N_2, m_1\rangle; A\ receives^i\{N_2, Q_2, k, m_2\}_{AT}^{s_3};$
$A\ sends^i m_2; A\ receives^i\{n_3\}_k^{s_4}; A\ sends^i\{n_3, A\}_k^{r_5}$

$Resp_{sNS}^B[T, B, i', Q_1, k', N_1, N_3, r_1, s_2, r_4, s_5] \equiv$
$B\ receives^{i'} Q_1; B\ generates^{i'} N_1; B\ sends^{i'}\{Q_1, N_1\}_{BT}^{r_1};$
$B\ receives^{i'}\{k', N_1, Q_1\}_{BT}^{s_2}; B\ generates^{i'} N_3;$
$B\ sends^{i'}\{N_3\}_{k'}^{r_4}; B\ receives^{i'}\{N_3, Q_1\}_{k'}^{s_5}$

$Trust_{sNS}^T[T, i'', Q_1', Q_2', K, n_1, n_2, s_1, r_2, r_3] \equiv$
$T\ receives^{i''}\langle Q_1', Q_2', n_2, \{Q_1', n_1\}_{Q_2'T}\rangle; T\ generates^{i''} K;$
$T\ sends^{i''}\{n_2, Q_2', K, \{K, n_1, Q_1'\}_{Q_2'T}^{r_2}\}_{Q_1'T}^{r_3}$

They further have to agree that $Q_1 = Q_1' = A$, $Q_2 = Q_2' = B$, $n_1 = N_1$, $n_2 = N_2$, $n_3 = N_3$, $k = k' = K'$.

Remark 2.6. Notice the variables m_1 and m_2 in the initiator's role. For example, the role says $A\ receives^i\{N_2, Q_2, k, m_2\}_{AT}^{s_3}$ instead of $A\ receives^i\{N_2, Q_2, k, \{k', N_1, Q_1\}_{BT}^{s_2}\}_{AT}^{s_3}$. The reason is, that A does not parse that part of the message, it does not look into the encryption, just forwards

m_2. In the course of the agreement proof, we show that $m_2 = \overline{\{k', N_1, Q_1\}^{s_2}_{BT}}$.

2.2. Agreement and authentication

The authentication and agreement properties that we consider have the following general pattern: An honest principal thinks he finished a session with certain other principals. Assuming that the others are also honest, we would like to prove that he really did communicate with the others, the others finished their parts, and the sent and received items that should be the same according the the protocol, were indeed the same.

For example, in case of the symmetric NS protocol, if the responder B finished a session with A and T, that is,

$$Resp^B_{sNS}[T, B, i', Q_1, k', N_1, N_3, r_1, s_2, r_4, s_5]$$

is satisfied, and the others are carrying out their roles honestly, and generate and encrypt honestly, then we want to show that the others did have a session with the responder, and they agree on the messages involved. We want to prove that for some i, i'', N_2, m_1, m_2, s_1, r_2, s_3, r_3, s_4, r_5,

$$Init^A_{sNS}[T, A, i, B, m_1, m_2, k', N_2, n_3, s_3, s_4, r_5],$$

$$Trust^T_{sNS}[T, i'', A, B, k', N_1, N_2, s_1, r_2, r_3]$$

are also satisfied.

We first need to formulate what it means to follow the roles honestly and not doing anything else. Let us introduce the following abbreviation:

$$\begin{aligned}
&Only_\Delta(A\ acts^i_1 t_1; ...; A\ acts^i_k t_k) \equiv \\
&\exists \tau_1 ... \tau_k \Big(A\ acts^i_{1,\tau_1} t_1; ...; A\ acts^i_{k,\tau_k} t_k \wedge \\
&\forall \tau'_1 ... \tau'_k t'_1 ... t'_k \Big(\big(A\ acts^i_{1,\tau'_1} t'_1; ...; A\ acts^i_{k,\tau'_k} t'_k \\
&\qquad\qquad \rightarrow \bigwedge_j \tau_j = \tau'_j \wedge t_j = t'_j \big) \\
&\qquad \bigwedge_{\substack{(acts_1, ..., acts_k) : \\ (acts_1, ..., acts_k) \neq \\ (acts_{k+1}, ..., acts_{2k})}} \neg A\ acts^i_{k+1,\tau'_1} t'_1; ...; A\ acts^i_{2k,\tau'_k} t'_k \Big) \Big) \Big|_\Delta
\end{aligned}$$

The meaning of this is that A in session i carries out these and only these actions. It is a consequence of this definition that for $\vec{\alpha}^{A,i}$, $\vec{\beta}^{A,i}$, if $\vec{\beta}^{A,i} \not\subseteq \vec{\alpha}^{A,i}$, then

$$Only_\Delta(\vec{\alpha}^{A,i}) \rightarrow \big(\vec{\alpha}^{A,i} \wedge \neg\, \vec{\beta}^{A,i} \big)\big|_\Delta$$

For $\vec{\alpha}^{A,i}_{\leq j} \equiv \alpha^{A,i}_1; ...; \alpha^{A,i}_n$ and $0 \leq j \leq n$, let $\vec{\alpha}^{A,i}_{\leq j}$ denote an initial segment of $\vec{\alpha}^{A,i}$ ending with $\alpha^{A,i}_j$, i.e.,

$$\vec{\alpha}^{A,i}_{\leq j} \equiv \alpha^{A,i}_1; \cdots; \alpha^{A,i}_j.$$

Let $Only_\Delta(\vec{\alpha}^{A,i}_{\leq 0}) \equiv \neg\exists t(A\ generates^i t \vee A\ sends^i t \vee A\ receives^i t)|_\Delta$ and $S = \{0, n\} \cup \{j | \exists t(\alpha^{A,i}_j = A\ sends^i t)\}$,

$$Foll(\vec{\alpha}^{A,i}) \equiv \forall \Delta \bigvee_{j \in S} Only_\Delta(\vec{\alpha}^{A,i}_{\leq j})$$

This formula means that A in session i does nothing but follow $\vec{\alpha}^{A,i}$, and may finish it, or stop after a send action, but not after a receive action. Finally, for $\alpha^{A,i}[A, i, \vec{m}]$, let

$$FOLL(\alpha^A) \equiv \forall i \exists \vec{m} Foll(\alpha^{A,i}[A, i, \vec{m}])$$

that is, A, in each of its sessions, follows α^A with some values. For example, a principal is carrying out the symmetric Needham-Schroeder protocol's initiator's role in each of his sessions and does nothing else, if

$$\begin{aligned}
&FOLL(Init^A_{\mathrm{NSL}}) \equiv \\
&\forall i \exists Q_2 m_1 m_2 k N_2 n_3 s_3 s_4 r_5 \\
&Foll(Init^A_{sNS}[T, A, i, Q_2, m_1, m_2, k, N_2, n_3, s_3, s_4, r_5])
\end{aligned}$$

In general, the authentication property from A's view has the following form:

$$Role^{A,i}[\vec{A}] \wedge \bigwedge_{B \sqsubseteq \vec{A}} FOLL(Role^B[\vec{Q}]) \rightarrow \bigwedge_{B \sqsubseteq \vec{A}} \exists i'\, Role^{B,i'}[\vec{Q}/\vec{A}]$$

Meaning that if principal A finishes his role playing with principals in $\vec{A}$ who honestly follow their roles playing with some principals $\vec{Q}$, then they all have a session that they carried out with the group $\vec{A}$. Agreement further requires some values of their roles to match. In case of the symmetric NS protocol, authentication together with agreement from the responder's view takes the form:

$$\begin{aligned}
&Resp^B_{sNS}[T, B, i', A, k, N_1, N_3, r_1, s_2, r_4, s_5] \\
&\wedge\ FOLL(Init^A_{\mathrm{sNS}}) \wedge FOLL(Resp^B_{\mathrm{sNS}}) \wedge FOLL(Trust^T_{\mathrm{sNS}}) \\
&\vdash \exists i i'' m_1 m_2 N_2 s_1 r_2 r_3 s_3 s_4 r_5 \qquad (1) \\
&\quad (Init^A_{\mathrm{sNS}}[T, A, i, B, m_1, m_2, k, N_2, N_3, s_3, s_4, r_5] \\
&\quad \wedge\ Trust^T_{sNS}[T, i'', A, B, k, N_1, N_2, s_1, r_2, r_3])
\end{aligned}$$

2.3. Secrecy Preservation

As we mentioned in the introduction, the central idea of our system is to prove first that the send actions of honest participants in a protocol do not corrupt the secrecy of nonces or keys. That is, if they were secret before the send action, then they are secret immediately after. Of course, some nonces and keys may be revealed and are revealed without any problem. For example, those nonces that were generated by an honest participant in a session that is played with a corrupted participant are of course allowed to be revealed. The first idea that comes to mind is that we prove this secrecy preservation for nonces or keys ν that satisfy some condition $C[\nu]$. Therefore, we want to prove is something like

$$\forall A i t \nu \tau \Delta \Big(A \sqsubseteq \vec{A} \ \wedge \ C[\nu] \ \wedge \ A \ sends^i_\tau t \ \wedge \ [Key] Sec_\tau(\vec{A}, \nu) \\ \longrightarrow \ [Key] Sec_\tau(\vec{A}, t, \nu) \Big) \Big|_\Delta$$

which is actually two properties, one with Key, one without it. This would mean that for each send action of the principals of the group $\vec{A}$, if any ν satisfying $C[\nu]$ is a secret of the group before the the send action, then it is not corrupted via the send action, as revealing the sent item maintains the secrecy of it. If we could prove this for a protocol, then it would ensure the secrecy of such ν's all the way trough as they are never revealed. However, this turns out to be normally impossible to prove. The property that we need is that if secrecy before the send action holds *even if we reveal certain other things*, then the secrecy is not corrupted with the send action. So accordingly, we define the following property: Let $C[\nu]$ be a condition on ν, and $C'[u]$ on term u.

$$[Key] SecSend(\vec{A}, C, C') \equiv \\ \forall A i t \nu u \tau \Delta \Big(A \sqsubseteq \vec{A} \wedge C[\nu] \wedge C'[u] \wedge \nu \not\sqsubseteq u \wedge A \ sends^i_\tau t \qquad (2) \\ \wedge \forall u' \Delta' \Big(\Delta' \subseteq \Delta \wedge C'[u'] \wedge \nu \not\sqsubseteq u' \rightarrow [Key] Sec_\tau(\vec{A}, u', \nu) \Big) \Big|_{\Delta'} \\ \longrightarrow [Key] Sec_\tau(\vec{A}, \langle t, u \rangle, \nu) \Big) \Big|_\Delta$$

Or, somewhat relaxed version is also sufficient:

$$[Key] SecSend(\vec{A}, \hat{C}) \equiv \forall A i t u \nu \tau \Delta \Big(A \sqsubseteq \vec{A} \wedge \hat{C}[\nu, u] \wedge A \ sends^i_\tau t \\ \wedge \forall \nu' u' \Delta' \Big(\Delta' \subseteq \Delta \wedge \hat{C}[\nu', u'] \rightarrow [Key] Sec_\tau(\vec{A}, u', \nu') \Big) \Big|_{\Delta'} \qquad (3) \\ \longrightarrow [Key] Sec_\tau(\vec{A}, \langle t, u \rangle, \nu) \Big) \Big|_\Delta$$

with $\hat{C}[\nu, u]$ a condition on ν and u.

Example 2.7. (Preservation of Secrecy in the Needham-Schroeder-Lowe protocol) For example, in case of the NSL protocol, where $\vec{A} = \langle A, B \rangle$, we can prove this property fixing C as

$$C[N] \equiv \exists i r n \big(A \ generates^i N; A \ sends^i \{N, A\}^r_B \\ \vee B \ generates^i N; B \ sends^i \{n, N, B\}^r_A \big)$$

and C' as

$$C'[u] \equiv \forall t (t \sqsubseteq u \rightarrow \exists m (t = m) \vee \exists t_1 t_2 (t = \langle t_1, t_2 \rangle) \\ \wedge \forall m \big(m \sqsubseteq u \rightarrow \exists i (A \ generates^i m \vee B \ generates^i m) \big)$$

Here, $C[N]$ expresses that N was generated by A and intended to B, or N was generated by B and intended to A. The formula $SecSend(N, u)$ will guarantee the preservation of the secrecy of such a nonce. Of course, if N does not satisfy these conditions, that is, if it was not intended to be between A and B, then we do not care about its secrecy. The first line of $C'[u]$ line expresses that the only function symbol that appears in u is the pairing, and the second line expresses that all bitstrings of u were generated by A or B. So we have that u has the form $u = \langle N_1, ..., N_j \rangle$, where N_1, ..., N_j were all generated by A or B. Then $N \not\sqsubseteq u$ expresses that $N \neq N_i$ for $1 \leq i \leq j$. So what $SecSend$ tells us is that for each message that was sent by A or B, if all N's that were intended to be between A and B, were the secret of A and B before a send action, even when the other nonces generated by A and B are revealed, then all such N's remain secret after sending the message.

Example 2.8. (Preservation of Secrecy in the symmetric Needham-Schroeder protocol) In case of the symmetric NS protocol, where $\vec{A} = \langle A, B, T \rangle$, we can prove for example, the preservation of secrecy of the shared key by fixing C as

$$C[K] \equiv \exists i'' n_1 n_2 r_2 r_3 (T \ generates^{i''} K; \\ T \ sends^{i''} \{n_2, B, K, \{K, n_1, A\}^{r_2}_{BT}\}^{r_3}_{AT})$$

And $C'[u]$ as

$$C'[u] \equiv \forall t (t \sqsubseteq u \rightarrow \exists m (t = m) \vee \exists t_1 t_2 (t = \langle t_1, t_2 \rangle)) \\ \wedge \forall m \big(m \sqsubseteq u \rightarrow \exists i (T \ generates^i m \vee B \ generates^i m) \big)$$

Here, too, $C[K]$ ensures that K is generated by T, and meant for A and B. With these definitions, we are able to prove $KeySecSend(\langle A, B, T \rangle, C, C')$.

Commitment problem. For the sake of discussing commitment but sticking to the symmetric NS protocol, let us now assume that we would want to prove the secrecy of K, and not the key secrecy. Since at a certain point, B sends the message $\{N_3\}^{r_4}_K$, and A sends $\{N_3, A\}^{r_5}_K$. From this on, K becomes distinguishable from another randomly generated key. In fact, already after sending $\{N_3\}^{r_4}_K$, K may already not be indistinguishable. So there is no hope to prove $SecSend$ in its previous form, for all send actions of A and B and T. Let

$$Reveal_\tau(\langle A, B, T \rangle, K)[N_3, r_4] \equiv B \ sends^{i'}_\tau \{N_3\}^{r_4}_K.$$

Then, instead of (2), we can prove

$$SecSend(\vec{A}, C, C', Reveal) \equiv \\ \forall A i t \nu u \tau \vec{m} \Delta \Big(A \sqsubseteq \vec{A} \wedge C(\nu) \wedge C'[u] \wedge \nu \not\sqsubseteq u \wedge A \ sends^i_\tau t \\ \wedge \forall \Delta' \Big(\Delta' \subseteq \Delta \wedge Reveal_{\tau'}(\vec{A}, \nu)[\vec{m}] \rightarrow \tau < \tau' \Big) \Big|_{\Delta'} \\ \wedge \forall u' \Delta' \Big(\Delta' \subseteq \Delta \wedge C'[u'] \wedge \nu \not\sqsubseteq u' \rightarrow [Key] Sec_\tau(\vec{A}, u', \nu') \Big) \Big|_{\Delta'} \\ \longrightarrow [Key] Sec_\tau(\vec{A}, \langle t, u \rangle, \nu) \Big) \Big|_\Delta .$$

2.4. The Axioms of Basic Protocol Logic 2.0

Finally, we present the axioms. We extend the usual first-order predicate logic with equality by adding the following sets of axioms.

(I) Axioms for the stochastic structure of the sets branches of coin tosses.

(1) Events

(a) $\Delta_1 \subseteq_\Delta \Delta_2 \wedge \Delta_2 \subseteq_\Delta \Delta_3 \rightarrow \Delta_1 \subseteq_\Delta \Delta_3$;

(b) $\Delta_1 \subseteq_\Delta \Delta_2 \wedge \Delta_1 \subseteq_\Delta \Delta_2 \leftrightarrow \Delta_1 =_\Delta \Delta_2$;

(c) $\Delta \subseteq \mathbf{D}$

(d) $\Delta_1 \subseteq \Delta \wedge \Delta_1 \subseteq_\Delta \Delta_2 \rightarrow \Delta_1 \subseteq \Delta_2$

(2) Time sections

(a) $\tau_1 <_\Delta \tau_2 \wedge \tau_2 <_\Delta \tau_3 \rightarrow \tau_1 <_\Delta \tau_3$;
$\neg(\tau_1 <_\Delta \tau_2 \wedge \tau_2 <_\Delta \tau_1)$;

(b) $\forall\tau(\mathbf{0} <_\Delta \tau)$;
$\forall\tau(\tau <_\Delta \mathbf{1} \rightarrow \tau =_\Delta \mathbf{0})$;
$\forall\tau(\tau <_\Delta \infty)$;

(c) $\big(\forall\Delta(\tau \not<_\Delta \tau' \wedge \tau' \not<_\Delta \tau) \rightarrow \tau = \tau'\big)\big|_{\Delta'}$

(II) Reflexibility and substitution properties of $=_\Delta$. They express that $=_\Delta$ has the usual properties of equation in first order logic when the predicates are restricted to Δ.

(1) $v =_\Delta v$;

(2) For any function symbol f,

$$v =_\Delta v' \rightarrow f(..., v, ...) =_\Delta f(..., v', ...)$$

(3) If φ_0 is any of the atomic formulas without $\big|_\Delta$, v is a variable in φ_0 and φ_0' is a formula that we get from φ_0 by replacing some occurrences of v by v', then

$$v =_\Delta v' \wedge \varphi_0\big|_\Delta \rightarrow \varphi_0'\big|_\Delta$$

(III) Restriction of formulas by $\big|_\Delta$. The meaning of (1) is that the atomic formulas are such that they keep their validity by restriction. For example, if two formulas are equal on Δ, they are equal on every subsets of Δ. If a term was sent on a set, it was sent also on every subset, etc. The meaning of (2) is, that if a formula φ keeps its validity by restriction, and if for all Δ, φ on Δ implies φ' on a subset of Δ, then we have that φ on Δ implies φ' on Δ. So it is enough to prove φ' on a subset. (The soundness proof of this is very simple: the largest subset of Δ, where φ' holds must be Δ, otherwise easily get a contradiction by making it even larger.)

(1) If φ_0 is an atomic formula without $\big|_\Delta$, then

$$\Delta' \subseteq \Delta \wedge \varphi_0\big|_\Delta \rightarrow \varphi_0\big|_{\Delta'}$$

(2) Let φ and φ' any formulas. Then

$$\begin{array}{l}\forall\Delta'\big(\Delta' \subseteq \Delta \wedge \varphi\big|_\Delta \rightarrow \varphi\big|_{\Delta'}\big) \\ \quad \wedge \big(\varphi\big|_\Delta \rightarrow \exists\Delta'(\Delta' \subseteq \Delta \wedge \varphi'\big|_{\Delta'})\big) \\ \longrightarrow \big(\varphi\big|_\Delta \rightarrow \varphi'\big|_\Delta\big)\end{array}$$

(IV) Term axioms. These are the axioms that result the examples we listed in Example 2.2. They also express that pairing is invertible, that a ciphertext decrypts to the correct plaintext by the correct key. They also express that correctly generated nonces, keys, encryptions cannot be the same random variables as other correctly generated items. Etc.

(1) Axioms for $=$ with not involving $\bar{\ }$

(a) $\langle t_1, t_2\rangle =_\Delta \langle t_1', t_2'\rangle \rightarrow t_1 =_\Delta t_1' \wedge t_2 =_\Delta t_2'$;

(b) $\{t\}^s_Q =_\Delta \{t'\}^{s'}_{Q'} \rightarrow t =_\Delta t' \wedge Q =_\Delta Q'$;
$\{t\}^s_{Q_1Q_2} =_\Delta \{t'\}^{s'}_{Q_1'Q_2'} \rightarrow t =_\Delta t' \wedge Q_1 =_\Delta Q_1' \wedge Q_2 =_\Delta Q_2'$;
$\{t\}^s_k =_\Delta \{t'\}^{s'}_{k'} \rightarrow t =_\Delta t' \wedge k =_\Delta k'$;

(c) $\{t\}^s_Q \neq_\Delta \langle t_1, t_2\rangle$; $\{t\}^s_Q \neq_\Delta m$; $\{t\}^s_Q \neq_\Delta \{t'\}^s_{Q_1Q_2}$; $\{t\}^s_{Q_1Q_2} \neq_\Delta \langle t_1, t_2\rangle$; $\{t\}^s_{Q_1Q_2} \neq_\Delta m$; $m \neq_\Delta \langle t_1, t_2\rangle$; $\{t\}^s_k \neq_\Delta \langle t_1, t_2\rangle$; $\{t\}^s_k \neq_\Delta m$

(d) $A \neq_\Delta N$; $A \neq_\Delta K$; $A \neq A' \rightarrow A \neq_\Delta A'$;

(e) $\{t\}^s_{Q_1Q_2} =_\Delta \{t\}^s_{Q_2Q_1}$;

(2) Axioms for $=$ with involving $\bar{\ }$

(a) $\overline{m} =_\Delta m$;

(b) For any function symbol f, $\overline{f(..., t, ...)} =_\Delta \overline{f(..., \bar{t}, ...)}$;

(c) $\bot \neq_\Delta t_1 \wedge \bot \neq_\Delta t_2 \rightarrow \bot \neq_\Delta \langle t_1, t_2\rangle$;
$\bot \neq_\Delta \overline{\langle t_1, t_2\rangle} =_\Delta \overline{\langle t_1', t_2'\rangle} \rightarrow \overline{t_1} =_\Delta \overline{t_1'} \wedge \overline{t_2} =_\Delta \overline{t_2'}$;
$\bot \neq_\Delta \overline{\{t\}^s_Q} =_\Delta \overline{\{t'\}^{s'}_Q} \rightarrow \overline{t} =_\Delta \overline{t'}$;
$\bot \neq_\Delta \overline{\{t\}^s_{Q_1Q_2}} =_\Delta \overline{\{t'\}^{s'}_{Q_1Q_2}} \rightarrow \overline{t} =_\Delta \overline{t'}$;
$\bot \neq_\Delta \overline{\{t\}^s_k} =_\Delta \overline{\{t'\}^{s'}_k} \rightarrow \overline{t} =_\Delta \overline{t'}$;

(d) $\overline{\{t\}^s_Q} =_\Delta \overline{\{t\}^{s'}_Q} \rightarrow \{t\}^s_Q =_\Delta \{t\}^{s'}_Q$;
$\overline{\{t\}^s_{Q_1Q_2}} =_\Delta \overline{\{t\}^{s'}_{Q_1Q_2}} \rightarrow \{t\}^s_{Q_1Q_2} =_\Delta \{t\}^{s'}_{Q_1Q_2}$;
$\overline{\{t\}^s_k} =_\Delta \overline{\{t\}^{s'}_k} \rightarrow \{t\}^s_k =_\Delta \{t\}^{s'}_k$;

(e) $\overline{\{t\}^r_A} \neq_\Delta A'$; $\overline{\{t\}^r_{A_1A_2}} \neq_\Delta A'$; $\overline{\{t\}^r_K} \neq_\Delta A'$;
$\big(N \neq \bot \rightarrow \overline{\{t\}^r_A} \neq N \wedge \overline{\{t\}^r_{A_1A_2}} \neq N \wedge \overline{\{t\}^r_K} \neq N\big)\big|_\Delta$;
$\big(K \neq \bot \rightarrow \overline{\{t\}^r_A} \neq K \wedge \overline{\{t\}^r_{A_1A_2}} \neq K \wedge \overline{\{t\}^r_{K'}} \neq K\big)\big|_\Delta$;

(f) For any compound term T containing only variables of sort `hseed`, `hname`, `hkey` or `hnonce`,
$\big((N \sqsubseteq T \wedge N \neq N') \vee K \sqsubseteq T \vee \{t\}^r_{...} \sqsubseteq T \rightarrow \overline{T} \neq N'\big)\big|_\Delta$;
$\big(N \sqsubseteq T \vee (K \sqsubseteq T \wedge K \neq K') \vee \{t\}^r_{...} \sqsubseteq T \rightarrow \overline{T} \neq K'\big)\big|_\Delta$;

(g) For any function symbol f, $\overline{f(..., \bot, ...)} =_\Delta \bot$;

(3) Axioms for $\sqsubseteq$

(a) $t_1 \sqsubseteq_\Delta t_2 \wedge t_2 \sqsubseteq_\Delta t_3 \rightarrow t_1 \sqsubseteq_\Delta t_3$;

(b) $t \sqsubseteq_\Delta t_1 \vee t \sqsubseteq_\Delta t_2 \rightarrow t \sqsubseteq_\Delta \langle t_1, t_2\rangle$;

(c) $t_1 \sqsubseteq_\Delta t_2 \rightarrow t_1 \sqsubseteq_\Delta \{t_2\}^s_Q$; $t_1 \sqsubseteq_\Delta t_2 \rightarrow t_1 \sqsubseteq_\Delta \{t_2\}^s_{Q_1Q_2}$; $t_1 \sqsubseteq_\Delta t_2 \rightarrow t_1 \sqsubseteq_\Delta \{t_2\}^s_k$;

(4) Axioms for $\sqsubseteq$ and $=$

(a) $t_1 =_\Delta t_2 \rightarrow t_1 \sqsubseteq_\Delta t_2$; $t_1 \sqsubseteq_\Delta t_2 \wedge t_2 \sqsubseteq_\Delta t_1 \rightarrow t_1 =_\Delta t_2$;

(b) $t \sqsubseteq_\Delta m \rightarrow t =_\Delta m$;

(c) $t \sqsubseteq_\Delta \langle t_1, t_2\rangle \rightarrow t \sqsubseteq_\Delta t_1 \vee t \sqsubseteq_\Delta t_2 \vee \langle t_1, t_2\rangle =_\Delta t$;

(d) $t_1 \sqsubseteq_\Delta \{t_2\}^s_Q \rightarrow t_1 \sqsubseteq_\Delta t_2 \vee \{t_2\}^s_Q =_\Delta t_1$;
$t_1 \sqsubseteq_\Delta \{t_2\}^s_{Q_1Q_2} \rightarrow t_1 \sqsubseteq_\Delta t_2 \vee \{t_2\}^s_{Q_1Q_2} =_\Delta t_1$;
$t_1 \sqsubseteq_\Delta \{t_2\}^s_k \rightarrow t_1 \sqsubseteq_\Delta t_2 \vee \{t_2\}^s_k =_\Delta t_1$;

(V) Trace with Secrecy: These axioms describe what can be revealed by still maintaining the secrecy of ν. They are used in the course of proving $[Key]SecSend$, that is, that send actions don't corrupt keys and nonces. In order to make the formulas less messy, we introduce a shorthand notation: $[Key]Sec_\tau(\vec{A}, t \xrightarrow{\varphi} t', \nu)|_\Delta$ means $\varphi|_\Delta \wedge [Key]Sec_\tau(\vec{A}, t, \nu)|_\Delta \rightarrow [Key]Sec_\tau(\vec{A}, t', \nu)|_\Delta$.

(a) Items that were not yet generated, cannot be already corrupted and cannot already corrupt:
$\big(Q\ generates^i_\tau \nu \wedge \tau' < \tau\big)\big|_\Delta \rightarrow [Key]Sec_{\tau'}(\vec{A}, \nu)\big|_\Delta$;
$\big(Q\ generates^i \nu \wedge \forall\Delta' \neg\exists t\tau\big(Q\ sends^i_\tau t \wedge \nu \sqsubseteq t \wedge \tau < \tau'\big)\big|_{\Delta'}$
$\longrightarrow [Key]Sec_{\tau'}(\vec{A}, \nu)\big)\big|_\Delta$;

$[Key]Sec_{\tau'}\big(\vec{A}, t \xrightarrow{Q\ generates^i_\tau \nu' \wedge \tau' < \tau} \langle t, \nu' \rangle, \nu\big)\big|_\Delta$;

$[Key]Sec_{\tau'}\Big(\vec{A}, t \xrightarrow{Q\, generates^i \nu' \wedge \neg \exists t \tau \binom{\nu' \sqsubseteq t \ \wedge\ \tau < \tau'}{Q\, sends^i_\tau t}} \langle t, \nu' \rangle, \nu\Big)\Big|_\Delta$;

(b) Names are public, so revealing them does not corrupt:
$[Key]Sec_\tau\big(\vec{A}, t \longrightarrow \langle t, Q \rangle, \nu\big)\big|_\Delta$;

(c) Revealing something t that was already revealed, shown by a send or receive action, cannot corrupt secrecy:
$\big(Q\, sends^i_\tau t \vee Q\, receives^i_\tau t\big)\big|_\Delta \wedge\ \tau <_\Delta \tau'$
$\wedge\, [Key]Sec_{\tau'}(\vec{A}, t', \nu)\big|_\Delta \rightarrow [Key]Sec_{\tau'}\big(\vec{A}, \langle t, t' \rangle, \nu\big)\big|_\Delta$;

(d) Something that is secret at time τ', was also secret earlier:
$\tau <_\Delta \tau' \wedge [Key]Sec_{\tau'}(\vec{A}, t, \nu)\big|_\Delta \rightarrow [Key]Sec_\tau\big(\vec{A}, t, \nu\big)\big|_\Delta$;

(e) Secrecy relative to different groups of principals:
$\forall A(A \sqsubseteq \vec{A} \rightarrow A \sqsubseteq \vec{A}') \wedge [Key]Sec_\tau\big(\vec{A}, t, \nu\big)\big|_\Delta$
$\rightarrow\ [Key]Sec_\tau\big(\vec{A}', t, \nu\big)\big|_\Delta$;
$[Key]Sec_\tau\big(\vec{A}, t, \nu\big)\big|_\Delta \wedge\ [Key]Sec_\tau\big(\vec{A}', t, \nu\big)\big|_\Delta$
$\rightarrow\ [Key]Sec_\tau\big(\langle \vec{A}, \vec{A}' \rangle, t, \nu\big)\big|_\Delta$;

(f) Manipulations of pairing inside the secrecy predicate:
$[Key]Sec_\tau\big(\vec{A}, \langle t, t' \rangle \longrightarrow t, \nu\big)\big|_\Delta$;
$[Key]Sec_\tau\big(\vec{A}, t \longrightarrow \langle t, t \rangle, \nu\big)\big|_\Delta$;
$[Key]Sec_\tau\big(\vec{A}, \langle t, t' \rangle \longrightarrow \langle t', t \rangle, \nu\big)\big|_\Delta$;
$[Key]Sec_\tau\big(\vec{A}, \langle \langle t, t' \rangle, t'' \rangle \longleftrightarrow \langle t, \langle t', t'' \rangle \rangle, \nu\big)\big|_\Delta$;

(g) ν reveals itself:
$\neg[Key]Sec_\tau\big(\vec{A}, \nu, \nu\big)\big|_\Delta$;

(h) Corruption depends only on the bit string, not how it was created:
$[Key]Sec_\tau\big(\vec{A}, t \longleftrightarrow \bar{t}, \nu\big)\big|_\Delta$;

(i) Honest encryptions of t' by secure keys cannot corrupt ν:
$[Key]Sec_\tau\big(\vec{A}, t \xrightarrow{A \sqsubseteq \vec{A}} \langle t, \{t'\}^r_A \rangle, \nu\big)\big|_\Delta$;
$[Key]Sec_\tau\big(\vec{A}, t, \xrightarrow{A \sqsubseteq \vec{A}\ \wedge\ A' \sqsubseteq \vec{A}} \langle t, \{t'\}^r_{AA'} \rangle, \nu\big)\big|_\Delta$;
$Sec_\tau\big(\vec{A}, t \xrightarrow{KeySec(\vec{A}, \langle t, t' \rangle, K) \wedge K \neq \nu} \langle t, \{t'\}^r_K \rangle, \nu\big)\big|_\Delta$;
$KeySec_\tau\big(\vec{A}, \langle t, t' \rangle \longrightarrow \langle t, \{t'\}^r_K \rangle, K\big)\big|_\Delta$;

(j) If t' does not corrupt ν, then its encryption does not either:
$[Key]Sec_\tau\big(\vec{A}, \langle t, t' \rangle \longrightarrow \langle t, \{t'\}^r_Q \rangle, \nu\big)\big|_\Delta$;
$[Key]Sec_\tau\big(\vec{A}, \langle t, t' \rangle \longrightarrow \langle t, \{t'\}^r_{QQ'} \rangle, \nu\big)\big|_\Delta$;
$[Key]Sec_\tau\big(\vec{A}, \langle t, t', k \rangle \longrightarrow \langle t, \{t'\}^r_k \rangle, \nu\big)\big|_\Delta$;

(k) If ciphertexts of corrupted encryptions don't corrupt ν, then the plaintext cannot either, as corrupted encryptions don't provide any secrecy:
$[Key]Sec_\tau\big(\vec{A}, \langle t, \{t'\}^s_Q \rangle \xrightarrow{Q \not\sqsubseteq \vec{A} \wedge \{t'\}^s_Q \neq \bot} \langle t, t', \{t'\}^s_Q \rangle, \nu\big)\big|_\Delta$;
$[Key]Sec_\tau\big(\vec{A}, \langle t, \{t'\}^s_{QQ'} \rangle \xrightarrow{Q \not\sqsubseteq \vec{A} \wedge \{t'\}^s_{QQ'} \neq \bot} \langle t, t', \{t'\}^s_{QQ'} \rangle, \nu\big)\big|_\Delta$;
$[Key]Sec_\tau\big(\vec{A}, \langle t, \{t'\}^s_K, K \rangle \xrightarrow{\{t'\}^s_K \neq \bot} \langle t, t', \{t'\}^r_K, K \rangle, \nu\big)\big|_\Delta$;

Remark 2.9. Key cycles. Note that this system takes into account that key cycles may corrupt security. The third item of V.j. has the condition $KeySec(\vec{A}, \langle t, t' \rangle, K)$ means that even if we reveal t' to the attacker, that is, it is free to submit it to the encryption oracle, the goodness of K is maintained. In a protocol that allows key cycles, this condition cannot be proven.

(VI) Axioms for relationship between properties. The following axioms connect actions, terms and cryptographic assumptions. Please note, that we do not put out $\big|_\Delta$ to the axioms just for better readability, unless the set for the premise and for the conclusion are different. Each axiom is meant to be limited to Δ.

(1) Actions and terms. The following axioms express that terms cannot be arbitrarily generated, sent.

(a) A term cannot be sent or received or generated and then generated:
$(\nu \sqsubseteq t \ \vee\ \{t'\}^s_\nu \sqsubseteq t)$
$\rightarrow \neg(Q_2\, sends^{i_2}/receives^{i_2}/generates^{i_1} t; Q_1\, generates^{i_1} \nu)$.

(b) Nothing can be generated in two different sessions or two different principals:
$(Q_1\ generates^{i_1} \nu_1\ \wedge\ Q_2\ generates^{i_2} \nu_2$
$\wedge\, (i_1 \neq i_2\ \vee\ Q_1 \neq Q_2))\ \rightarrow\ \nu_1 \neq \nu_2$.

(c) Something that was generated and then received by someone, had to be sent in the same session as it was generated, before receiving it:
$(\nu \sqsubseteq t_2\ \vee\ \{t\}^s_\nu \sqsubseteq t_2)\ \wedge\ Q\ generates^{i_1} \nu; Q'\ receives^{i_2} t_2$
$\rightarrow\ \exists t_1 t' s'((\nu \sqsubseteq t_1\ \vee\ \{t'\}^s_\nu \sqsubseteq t_1)$
$\wedge\, Q\ sends^{i_1} t'; Q'\ receives^{i_2} t)$

(d) Something that was sent by an honest agent had to be either generated by him, or received from someone else:
$(m \sqsubseteq t_2\ \vee\ \{t\}^s_m \sqsubseteq t_2)\ \wedge\ A\ sends^i t_2$
$\rightarrow\ (A\ generates^i m; A\ sends^i t_2$
$\vee\, \exists t_1 (A\ receives^i t_1; A\ sends^i t_2\ \wedge\ m \sqsubseteq t_1))$.

(e) Something on which an honest action was carried out, cannot be the error $\bot$. $(t \sqsubseteq t' \vee t = A) \wedge (A\ sends^i t' \vee A\ receives^i t' \vee A\ generates^i t') \rightarrow t \neq \bot$

(f) All honest nonces and keys were sometime generated:
$\exists A i \Delta(A\ generates^i N|_\Delta)$

(2) Secrecy. This axioms expresses that if no send action corrupted ν, then it had to remain uncorrupted. The axiom specified here is not suitable if reveal action is allowed. In that case security will hold until the reveal action (in Appendix). For $\vec{A} = \langle A_1, ..., A_n \rangle$ we postulate
$[Key]SecSend(\vec{A}, C, C')\ \wedge\ \big(C[\nu] \wedge C'[u] \wedge \nu \not\sqsubseteq u\big)\big|_\Delta$
$\rightarrow\ [Key]Sec_\infty(\vec{A}, u, \nu)\big|_\Delta$

(3) Authentication with public key. This says that if a message of the form $\{t'\}^s_A$ was received, and if $\{t'\}^s_A$ does not corrupt ν, but t' does, then, at least on a part of the traces (the part on which t' corrupts ν), $\{t'\}^s_A$ had to be created by one of the principals in $\vec{A}$ as only they may know ν. This axiom is sound as long as the public-key encryption satisfies IND-CCA security:
$\Big(\{t'\}^s_A \sqsubseteq t_2\ \wedge\ A\ receives^{i_2}_{\tau_2} t_2\ \wedge \tau \not< \tau_2$
$\wedge\, [Key]Sec_\tau(\vec{A}, \langle t, \{t'\}^s_A \rangle, \nu) \wedge \neg[Key]Sec_\tau(\vec{A}, \langle t, t' \rangle, \nu)\Big)\Big|_\Delta$
$\rightarrow\ \exists A' t_1 t'' i_1 r \tau_1 \Delta'\Big(A' \sqsubseteq \vec{A}\ \wedge\ \Delta' \subseteq \Delta\ \wedge\ \{t''\}^r_A \sqsubseteq t_1$
$\wedge\, \overline{\{t\}^s_A} = \overline{\{t''\}^r_A} \wedge \Big(A'\ sends^{i_1}_{\tau_1} t_1; A\ receives^{i_2}_{\tau_2} t_2\Big)\Big|_{\Delta'}\Big)$

(4) Authentication with long-term shared key: This axiom expresses the unforgeability (see section 3) of the shared key encryption for given plaintext. That is, if $\{t'\}^r_{AB}$ was received, and if it contains a randomly generated nonce or key,

then it had to be created and sent either by A or B:
$A\ receives^{i_2}_{\tau_2} t_2 \ \wedge\ \{t\}^s_{AB} \sqsubseteq t_2 \ \wedge\ N \sqsubseteq t \ \vee\ K' \sqsubseteq t$
$\rightarrow \exists A' i_1 t_1 t' ((A' = A \ \vee\ A' = B)$
$\wedge\ A'\ sends^{i_1}_{\tau_1} t_1; A\ receives^{i_2}_{\tau_2} t_2 \wedge \{t'\}^r_{AB} \sqsubseteq t_1$
$\wedge\ \overline{\{t\}^s_{AB}} = \overline{\{t'\}^r_{AB}}))$

(5) Authentication with shared session key

(a) This is the same as the previous for an uncorrupted key K:
$KeySec_{\tau_2}(\vec{A}, K) \wedge A\ receives^{i_2}_{\tau_2} t_2 \ \wedge\ \{t\}^s_K \sqsubseteq t_2$
$\wedge\ N \sqsubseteq t \ \vee\ K' \sqsubseteq t$
$\rightarrow \exists A' i_1 t_1 t' r \tau_1 (A' \sqsubseteq \vec{A} \wedge A'\ sends^{i_1}_{\tau_1} t_1; A\ receives^{i_2}_{\tau_2} t_2$
$\wedge\ \{t'\}^r_K \sqsubseteq t_1 \ \wedge\ \overline{\{t\}^s_K} = \overline{\{t'\}^r_K})$

(b) This axiom is similar to the one for public key. It says that if $\{t'\}^s_K$ does not corrupt ν but t' does, then, at least on a part of the traces, $\{t'\}^s_K$ had to be created by one of the principals in $\vec{A}$. Note, that this axiom does not require that K is uncorrupted. Nevertheless, soundness needs the IND-CCA property of the encryption.
$\Big(A \sqsubseteq \vec{A} \wedge \{t'\}^s_K \sqsubseteq t_2\ A\ receives^{i_2}_{\tau_2} t_2 \wedge \tau \not< \tau_2 \wedge [Key] Sec_\tau(\vec{A}, \langle t, \{t'\}^s_K\rangle, \nu) \wedge \neg [Key] Sec_\tau(\vec{A}, \langle t, t'\rangle, \nu))\Big)\Big|_\Delta$
$\rightarrow \exists A' i_1 t_1 t'' r \tau_1 \Delta' \Big(A' \sqsubseteq \vec{A} \wedge \Delta' \subseteq \Delta \wedge \Big(\{t''\}^r_K \sqsubseteq t_1$
$\wedge\ \overline{\{t\}^s_K} = \overline{\{t''\}^r_K} \wedge A'\ sends^{i_1}_{\tau_1} t_1; A\ receives^{i_2}_{\tau_2} t_2\Big)\Big|_{\Delta'}\Big)$

Let the presented theory be called $\mathcal{T}$. Let $\mathcal{T}'$ be the theory where all elements of sort `event` are replaced by $\mathbf{D}$, which practically means deleting all items of this sorts from every formulas. Then, $\mathcal{T}'$ is of course not sound, but it can be used to prove the agreement and authentication formulas as we have the following theorem:

Theorem 2.10. If φ is a formula without $|_\Delta$ satisfying $\forall \Delta' \big(\Delta' \subseteq \Delta \wedge \varphi\big|_\Delta \rightarrow \varphi\big|_{\Delta'}\big)$, that is, φ is restrictable, then $\varphi \vdash_{\mathcal{T}} \varphi'$ if and only if $\varphi \vdash_{\mathcal{T}'} \varphi'$.

Since the premisses of agreement and authentication properties are restrictable, we don't have to consider Δ's when we prove them. The proof of this theorem uses the fact that all of our axioms have the following pattern: If some formula is true on a Δ, another is true on a $\Delta' \subseteq \Delta$. So all derived formulas have this form as well, and then axiom III.2. proves the theorem.

3. Computational Assumptions

We need to fix a couple of computational operations: Pairing is an injective *pairing function* $[\cdot,\cdot]$: strings $\times$ strings $\rightarrow$ strings. We assume that the length of the pair only depends on the lengths of its inputs. We assume that the length of the ciphertext of encryptions depends only on the length of the plaintext. Honest nonces all have identical lengths as well as honest keys.

In this paper, we assume that the encryption schemes satisfie adaptive chosen ciphertext security (IND-CCA) both in case of asymmetric and symmetric encryptions. For the definition of IND-CCA security, we refer the reader to [3].

Unforgeability. In addition to IND-CCA security, we require that the symmetric encryption satisfies an unforgeability condition. Clearly, agreement in the symmetric NS protocol cannot possibly be proven from the initiator's view, if the encryption $\{n_3\}^s_K$ can be forged. In that case, the initiator cannot be sure where this message came from. To prevent this, we need to require the so-called existential unforgeability (see [15]), saying that no encryption for a given secret key can be forged. In this case however, an encryption cannot possibly be confused with another encryption, so there is unarbitrary parsing for this encryption. We would like to emphasize though that our method works even when parsing in general may be arbitrary. So we we do not assume this property (but they can easily be modified for it), and we don't prove agreement for sNS in initiator's view. Instead, we assume random plaintext unforgeability (see again [15]) but allowing decryption oracle for the attacker. In this case, the attacker has to create a ciphertext for given, random plaintext while allowed to quarry encryption and decryption oracles. The authenticity of $\{n_3\}^s_K$ cannot be ensured, but of all other messages can, so we can prove agreement from the responder's view.

4. Amended Needham-Schroeder Authentication and Agreement

In this section, we would like to illustrate how the proof works on the Amended Needham-Schroeder protocol introduced in Example 2.5. The agreement - authentication property was discussed earlier, it is expressed by the sequent (1). That is, given that the responder finished its role, and that the trusted party and the initiator follow their roles honestly, we can conclude that the initiator and the trusted party also finish their roles and the nonces and keys match. This is proven via first proving $KeySecSend$:

$FOLL(Init^A_{sNS}) \ \wedge\ FOLL(Resp^B_{sNS}) \ \wedge\ FOLL(Trust^T_{sNS})$
$\vdash\ KeySecSend(\langle A, B, T\rangle, C, C').$

This amounts to proving that

$FOLL(Init^A_{sNS}) \ \wedge\ FOLL(Resp^B_{sNS}) \ \wedge\ FOLL(Trust^T_{sNS})$
$\wedge\ C[K] \ \wedge\ C'[u] \ \wedge\ K \not\sqsubseteq u \wedge A' \sqsubseteq \langle A, B, T\rangle \ \wedge\ A'\ sends^i_\tau t$
$\wedge\ \forall K' u' \big(C[K'] \ \wedge\ C'[u'] \ \wedge\ K' \not\sqsubseteq u'$ (4)
$\rightarrow\ KeySec_\tau\big(\langle A, B, T\rangle, u', K'\big)\big)$
$\vdash KeySec_\tau\big(\langle A, B, T\rangle, \langle t, u\rangle, K\big).$

C and C' were defined in Example 2.8.

For limited space, we just indicate the idea of the proof. By Theorem 2.10, we don't need to use Δ's in the proof.

Proof sketch of (4)**.** In the premise, the possibilities of what $A'\ sends^i_\tau t$ can be is limited, as A' may only be either T, A, or B, and they all follow the protocol roles honestly. So the possibilities are the following:
1.) If $A' = T$, then for some $n'_1, n'_2, Q'_1, Q'_2, K', r_1, r'_2$,

$$t = \{n'_2, Q'_2, K', \{K', n'_1, Q'_1\}^{r'_2}_{Q'_2 T}\}^{r'_3}_{Q'_1 T}.$$

2.) If $A' = B$ then for some $Q_1, N_1, N_3, k, r_1, r_4$,

$$t = \{Q_1, N_1\}^{r_1}_{BT} \ \vee \ t = \{N_3\}^{r_4}_{k}.$$

3.) If $A' = A$, then for some $Q_2, N_2, m_1, m_2, n_3, r_5, k$,

$$t = A \vee t = \langle A, Q_2, N_2, m_1\rangle \vee t = m_2 \vee t = \{n_3, A\}^{r_5}_{k}.$$

In the proof, we go through all these cases, and show that $KeySec_\tau\big(\langle A, B, T\rangle, \langle t, u\rangle, K\big)$ holds. Here there is no room to do that, we only show one specific case, when

3.3.) $t = m_2$. So we have to show

$$KeySec_\tau\big(\langle A, B, T\rangle, \langle m_2, u\rangle, K\big), \tag{5}$$

which means tracking down where m_2 came from, what it is equal to, and so show that it does not corrupt the security of K. Since A follows his role honestly, that is $FOLL(Init^A_{\mathrm{sNS}})$ holds, we have

$$A\ receives^i_{\tau'}\{N_2, Q_2, k, m_2\}^{s_3}_{AT}; A\ sends^i_\tau m_2. \tag{6}$$

That is, A received m_2 in a secure encryption from the trusted party T. Axiom VI.4. tells us, that the long term shared keys of honest participants guarantee authentication, that is, the message had to come from T:

$$\begin{aligned}\exists A' i'' t_1 t' ((A'' = A \ \vee \ A' = T) \ \wedge \ \{t'\}^r_{AT} \sqsubseteq t_1 \\ \wedge \ \overline{\{N_2, Q_2, k, m_2\}^{s_3}_{AT}} = \overline{\{t'\}^r_{AT}} \\ \wedge \ A''\ sends^{i''}_{\tau_1} t_1; A\ receives^i_{\tau'}\{N_2, Q_2, k, m_2\}^{s_3}_{AT}))\end{aligned} \tag{7}$$

and the third conjunct implies that

$$\overline{\langle N_2, Q_2, k, m_2\rangle} = \overline{t'}. \tag{8}$$

$A\ sends^{i''} t_1 \wedge \{t'\}^r_{AT} \sqsubseteq t_1$ is not possible because of $FOLL(Init^A_{sNS})$ and term axioms (A never sends out a message using his shared key with T). So $A'' = T$. But $T\ sends^{i''} t_1$ and $FOLL(Trust^T_{sNS})$ imply that $t_1 = \{n_2, Q'_2, K, \{K, n_1, Q'_1\}^{r_2}_{Q'_2T}\}^{r_3}_{Q'_1T}$. Then $\{t'\}^r_{AT} \sqsubseteq t_1$ implies with term axioms that

$$\begin{aligned}\{t'\}^r_{AT} &= \{K, n_1, Q'_1\}^{r_2}_{Q'_2T} \vee \\ \{t'\}^r_{AT} &= \{n_2, Q'_2, K, \{K, n_1, Q'_1\}^{r_2}_{Q'_2T}\}^{r_3}_{Q'_1T}.\end{aligned}$$

3.3.1.) If $\{t'\}^r_{AT} = \{K, n_1, Q'_1\}^{r_2}_{Q'_2T}$, then $Q'_2 = A$, and

$$t' = \langle K, n_1, Q'_1\rangle. \tag{9}$$

By (8) and (9), $\overline{\langle N_2, Q_2, k, m_2\rangle} = \overline{\langle K, n_1, Q'_1\rangle}$, that is, by our convention, $\overline{\langle\langle N_2, Q_2\rangle, k\rangle, m_2\rangle} = \overline{\langle\langle K, n_1\rangle, Q'_1\rangle}$, which, by term axioms, implies that $\overline{\langle N_2, Q_2\rangle} = K$, which results in a contradiction, because we would have

$$True \equiv Sec_{\mathbf{0}}(\vec{A}, K, N_2) \leftrightarrow Sec_{\mathbf{0}}(\vec{A}, \langle N_2, Q_2\rangle, N_2) \equiv False.$$

So $\{t'\}^r_{AT} \neq \{K, n_1, Q'_1\}^{r_2}_{Q'_2T}$.

3.3.2.) If $\{t'\}^r_{AT} = \{n_2, Q'_2, K, \{K, n_1, Q'_1\}^{r_2}_{Q'_2T}\}^{r_3}_{Q'_1T}$, then by term axioms, $Q'_1 = A$, and

$$t' = \langle n_2, Q'_2, K, \{K, n_1, A\}^{r_2}_{Q'_2T}\rangle. \tag{10}$$

By (8) and (10), $\overline{\langle n_2, Q'_2, K, \{K, n_1, A\}^{r_2}_{Q'_2T}\rangle} = \overline{\langle N_2, Q_2, k, m_2\rangle}$, and by term axioms,

$$N_2 = n_2 \ \wedge \ Q_2 = Q'_2 \ \wedge \ K = k \ \wedge m_2 = \overline{\{K, n_1, A\}^{r_2}_{Q'_2T}}.$$

So we know that $m_2 = \overline{\{K, n_1, A\}^{r_2}_{Q'_2T}}$, and so to complete the proof of (5), by axiom V.h., we have to show that

$$KeySec_\tau\big(\langle A, B, T\rangle, \langle\{K, n_1, A\}^{r_2}_{Q'_2T}, u\rangle, K\big).$$

From (6), (7), and that T follows the protocol,

$$\begin{aligned}&T\ receives^{i''}_{\tau''_1}\langle A, Q_2, N_2, \{A, n_1\}_{Q_2T}\rangle; T\ generates^{i''}_{\tau'_1} K'; \\ &T\ sends^{i''}_{\tau_1}\{N_2, Q_2, K', \{K', n_1, A\}^{r_2}_{Q_2T}\}^{r_3}_{AT}; \\ &A\ receives^i_{\tau'}\{N_2, Q_2, K', m_2\}^{s_3}_{AT}; A\ sends^i_\tau m_2\end{aligned} \tag{11}$$

follows with $k = K'$ and $m_2 = \overline{\{K', n_1, A\}^{r_2}_{Q_2T}}$, where $Q_2 \neq T$.

3.3.1.) If $Q_2 \neq B \ \wedge \ Q_2 \neq A$, then K' does not satisfy C, so $K \neq K'$. But $T\ generates^{i''} K'$, so $\langle K', u\rangle$ satisfies C', and therefore, by the premise of $SecSend$,

$$KeySec_\tau(\langle A, B, T\rangle, \langle K', u\rangle, K).$$

Since $\tau''_1 < \tau$, and $T\ receives^{i''}_{\tau''_1}\langle A, Q_2, N_2, \{A, n_1\}_{Q_2T}\rangle$, by axiom V.c., we have with $\vec{A} = \langle A, B, T\rangle$,

$$KeySec_\tau(\vec{A}, \langle\langle A, Q_2, N_2, \{A, n_1\}_{Q_2T}\rangle, K', u\rangle, K)$$

by this and ax. V.f., $KeySec_\tau(\vec{A}, \langle\{A, n_1\}_{Q_2T}, K', u\rangle, K)$, which in turn implies by axiom V.k., that $KeySec_\tau(\vec{A}, \langle A, n_1, \{A, n_1\}_{Q_2T}, K', u\rangle, K)$. By IV.f again, $KeySec_\tau(\vec{A}, \langle K', n_1, A, u\rangle, K)$. So, by axiom V.j.,

$$KeySec_\tau(\langle A, B, T\rangle, \langle\{K', n_1, A\}^{r_2}_{Q_2T}, u\rangle, K).$$

3.3.2.) If $Q_2 = B \ \vee \ Q_2 = A$, then by Ax V.i,

$$KeySec_\tau(\langle A, B, T\rangle, \langle\{K', n_1, A\}^{r_2}_{BT}, u\rangle, K),$$

so $KeySec_\tau(\langle A, B, T\rangle, \langle m_2, u\rangle, K)$ was shown in all cases.

In a similar manner, we can prove that none of the other send actions of A, B and T in any session corrupt K.

Proof sketch of (1)**.** The premise of this formula, namely, the responder's role implies $B\ receives^{i'}\{k, N_1, A\}^{s_2}_{BT}$. The same way as we proved (11), we can show that k had to be generated and sent by T. So there is a $K = k$, with

$$\begin{aligned}&T\ generates^{i''} K; T\ sends^{i''}\{n_2, B, K, \{K, N_1, A\}^{s_2}_{BT}\}^{r_3}_{AT}; \\ &B\ receives^{i'}\{K, N_1, A\}^{s_2}_{BT}; B\ receives^{i'}\{N_3, A\}^{s_5}_{K}.\end{aligned}$$

Therefore, we have

$$\exists i'' n_2 s_1 r_2 r_3\, Trust^T_{sNS}[T, i'', A, B, k, N_1, n_2, s_1, r_2, r_3].$$

Also, K satisfies C. Hence, since we showed $KeySecSend$, by axiom VI.2. we have $KeySec_\infty(\langle A, B, T\rangle, K)$. Then,

$$\exists A' i_1 t_1 t' r(A' \sqsubseteq \langle A, B, T\rangle \wedge \{t'\}^r_K \sqsubseteq t_1 \wedge \overline{\{N_3, A\}^{s5}_K} = \overline{\{t'\}^r_K} \wedge A'\ sends^{i_1} t_1; B\ receives^{i'} \{N_3, A\}^{s5}_K).$$

By Axiom VI.5.a.

1.) $A' = T$ is not possible, because of $FOLL(Trust^T_{sNS})$: T never encrypts with session key.

2.) If $A' = B$, then by $FOLL(Resp^B_{sNS})$, $t_1 = \{N'_3\}^{r'_4}_K$ and $\overline{\{N'_3\}^{r'_4}_K} = \overline{\{N_3, A\}^{s5}_K}$, therefore $N'_3 = \overline{\langle N_3, A\rangle}$, which is not possible by Axiom IV.2.f., if $N_3 \neq N'_3$, and not possible by our additional assumption in Example 2.5 if $N_3 = N'_3$. So $A' \neq B$.

3.) If $A' = A$, then by $FOLL(Init^A_{\mathrm{sNS}})$, $t_1 = \{n_3, A\}^{r5}_K$, and by term axioms, $n_3 = N_3$, so

$$\exists i r_5(A\ sends^i \{N_3, A\}^{r5}_K; B\ receives^{i'} \{N_3, A\}^{s5}_K).$$

So by $FOLL(Init^A_{\mathrm{sNS}})$,

$$\exists i Q_2 m_1 m_2 N_2 s_3 s_4 r_5\ Init^A_{\mathrm{sNS}}[T, A, i, Q_2, m_1, m_2, k, N_2, N_3, s_3, s_4, r_5].$$

In particular, $A\ receives^i \{N_2, Q_2, K, m_2\}^{r3}_{AT}$; $A\ sends^i m_2$; $A\ receives^i \{N_3\}^{s4}_K$; $A\ sends^i \{N_3, A\}^{s5}_K$. Just as the way we received (11), we have that

$$\exists i''' K n'_1 r'_2 r'_3 (m_2 = \overline{\{K, n'_1, A\}^{r'_2}_{Q_2 T}} \wedge T\ sends^{i'''} \{N_2, Q_2, K, \{K, n'_1, A\}^{r'_2}_{Q_2 T}\}^{r'_3}_{AT}; A\ receives^i \{N_2, Q_2, K, m_2\}^{r3}_{AT}; A\ sends^i m_2; A\ receives^i \{N_3\}^{s4}_K; A\ sends^i \{N_3, A\}^{s5}_K).$$

But since by $FOLL(Trust^T_{\mathrm{sNS}})$, $T\ generates^{i''} K \wedge T\ generates^{i'''} K$, according to axiom VI.1.b, we have $i'' = i'''$. Then by $FOLL(Trust^T_{\mathrm{sNS}})$,

$$\{N_2, Q_2, K, \{K, n'_1, A\}^{r'_2}_{Q_2 T}\}^{r'_3}_{AT} = \{n_2, B, K, \{K, N_1, A\}^{s2}_{BT}\}^{r3}_{AT}$$

and term axioms, $n'_1 = N_1$, $n_2 = N_2$, $Q_2 = B$. Therefore,

$$\exists i i'' m_1 N_2 s_1 r_2 r_3 s_3 s_4 r_5\ (Init^A_{\mathrm{sNS}}[A, T, i, B, , m_1, m_2/\overline{\{k, N_1, A\}^{r2}_{BT}}, k, N_2, N_3, s_3, s_4, r_5] \wedge Trust^T_{sNS}[T, i'', A, B, k, N_1, N_2, s_1, r_2, r_3]).$$

5. Discussion

Other results. We have also been able to prove agreement and authentication from the initiator's view of the sNS protocol when the axioms are modified for existential unforgeability. With the current set of axioms, we have proven agreement and authentication from both responder's and initiator's view.

Tuples. Note, that according to our choice, tuples of messages are created via pairings from left to right. If this convention is changed, the proofs have to be changed as well. There is no a priori guarantee that bracketing in a different way will not result in new type-flaw attacks.

Type flaw attacks. In the proof, we needed the additional axiom that $\langle N_3, A\rangle \neq N_3$. The assumptions that we presented for the computational pairing in fact prevent this (because of the length preservation and invertability, the left must be longer), but we did not include this in the set of axioms, as with length-hiding encryption, this assumption on the pairing is not needed. Allowing $\langle N_3, A\rangle = N_3$, there is an attack.

Honest names were assumed to be generated honestly, so this analysis does not account for attacks that use incorrectly assigned honest names. If we want to allow the possibility of such type-flaw attacks (see [5]), then we have to remove some of the term axioms from III.2.f and III.1.e, and do the proofs without them.

In the course of the proof on NSL protocol, we also need an additional assumption, namely, $\langle n, Q\rangle \neq N$. This is also prevented by the length assumption as long as malicious nonces must have fixed length too, and so introducing a new type, `nonce` for possibly malicious nonces, this assumption makes computational sense. Without this assumption however, there is an attack:

A type flaw attack on NSL. Take a Q such that for any value of N, there is an n such that $\langle N_3, A\rangle = N_3$. Catch $\{N_2\}_B$ and start a new session with B with this message. B will think he received $\{n, Q\}_B$, and will reveal n to Q in his response. So Q learns N_2.

Long-term keys. Observe, that in the current paper we assume that the long-term keys were honestly generated, and were successfully distributed earlier, after which they are not being sent around. This analysis cannot account for attacks that assume otherwise, for example, assume previously corrupted but replaced long-term keys.

6. Conclusions and future work

We have presented a relatively simple first-order proof system to analyze secrecy, agreement and authentication of protocols. Despite its simplicity, it takes care of key cycles, type-flaw issues and commitment problem. No idealizations are required for computational soundness. We have presented a theorem that tells that for the aimed properties, proof within a simpler (not sound) system is sufficient.

For future work, we are planning to introduce some lemmas that follow from the axioms to make the proofs simpler. We also plan to verify additional protocols. We want to introduce the possibility of multiple long term keys to be able to handle corrupted, replaced long term keys.

References

[1] M. Backes, B. Pfitzmann, and M. Waidner. A composable cryptographic library with nested operations. In *Proceedings of CCS'03*, pages 220–230. ACM Press, 2003.

[2] 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 33–58. Springer-Verlag, 2009.

[3] M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 531–545. Springer-Verlag, 2000.

[4] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. Technical report, Technical Report 39, Digital Systems Research Center, 1989.

[5] P. Ceelen, S. Mauw, and S. Radomirović. Chosen-name attacks: An overlooked class of type-flaw attacks. *Electron. Notes Theor. Comput. Sci.*, 197(2):31–43, 2008.

[6] I. Cervesato, N.A. Durgin, P.D. Lincoln, J.C. Mitchell, and A. Scedrov. Multiset rewriting and the complexity of bounded security protocols. *Journal of Computer Security*, 12(1):677–722, 2004.

[7] J. Clark and J. Jacob. A survey of authentication protocol literature: Version 1.0, 1997.

[8] Hubert Comon-Lundh and Véronique Cortier. Computational soundness of observational equivalence. In *ACM Conference on Computer and Communications Security*, pages 109–118, 2008.

[9] V. Cortier and B. Warinschi. Computationally sound, automated proofs for security protocols. In M. Sagiv, editor, *Proceedings of the 14th European Symposium on Programming (ESOP)*, volume 3444 of *LNCS*, pages 157–171, Edinburgh, UK, April 4–8 2005. Springer.

[10] A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. A derivation system and compositional logic for security protocols. *Journal of Computer Security*, 13:423–482, 2005.

[11] A. Datta, A. Derek, J. C. Mitchell, V. Shmatikov, and M. Turuani. Probabilistic polynomial-time semantics for a protocol security logic. In *Proceedings of ICALP'05*, volume 3580 of *LNCS*, pages 16–29. Springer, 2005.

[12] D. Dolev and A. C. Yao. On the security of public-key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, March 1983. Preliminary version presented at FOCS'81.

[13] N.A. Durgin, J.C. Mitchell, and D. Pavlovic. A compositional logic for proving security properties of protocols. *Journal of Computer Security*, 11:677–721, 2003.

[14] F. Javier Thayer Fábrega. Strand spaces: proving security protocols correct. *J. Comput. Secur.*, 7(2-3):191–230, 1999.

[15] J. Katz and M. Yung. Unforgeable encryption and chosen ciphertext secure modes of operation. In *Proceedings of the 7th International Workshop on Fast Software Encryption (FSE 2000)*, volume 1978 of *LNCS*, pages 25–36. Springer-Verlag, 2001.

[16] R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978.

[17] R. Needham and M. Schroeder. Authentication revisited. *ACM SIGOPS Operating Systems Review*, 21(1):7, 1987.

[18] A. Roy, A. Datta, A. Derek, and J. C. Mitchell. Inductive proofs of computational secrecy. In *ESORICS 2007*, volume 4734 of *LNCS*, pages 219–234, 2007.

[19] A. Roy, A. Datta, J. C. Mitchell, and J.P. Seifert. Secrecy analysis in protocol composition logic. In *11th Annual Asian Computing Science Conference (ASIAN'06)*, volume 4435 of *LNCS*, 2006.

A. Computational semantics

A.1. Elements of the computational execution

In this paper, there is no room for providing a detailed description of the semantics, so we briefly mention the assumptions that we need for computational soundness. We write a bit more about the semantics in the appendix.

First, we would like to remind the reader, that the fundamental objects of the computational world are bit strings, $\mathsf{strings} = \{0,1\}^*$, and – as the algorithms we consider are probabilistic – random variables of bit strings. Furthermore, hardness of computation is modeled by complexity theory, hence a size parameter, which is called *security parameter* in the context of cryptography is given as input to all algorithms considered here. Consequently, the probability fields and random variables resulting from the executions of the probabilistic algorithms are indexed by the security parameter. As usual, the security parameter is a natural number $\eta \in \mathbb{N}$, which, as an input of the algorithms, is represented as an η long string of 1's.

Definition A.1 (Negligible Function)**.** A function $f : \mathbb{N} \to \mathbb{R}$ is said to be *negligible*, if for any $c > 0$, there is an $n_c \in \mathbb{N}$ such that $|f(\eta)| \leq \eta^{-c}$ whenever $\eta \geq n_c$.

Pairing is an injective *pairing function* $[\cdot,\cdot] : \mathsf{strings} \times \mathsf{strings} \to \mathsf{strings}$. We assume that the length of the pair only depends on the lengths of its inputs. An asymmetric encryption scheme is a triple of algorithms $(\mathcal{K}_a, \mathcal{E}_a, \mathcal{D}_a)$ with probabilistic key generation $\mathcal{K}_a$, probabilistic encryption $\mathcal{E}_a$ and deterministic decryption $\mathcal{D}_a$; the key generation on input η outputs a random encryption-key decryption-key pair. $\mathcal{E}_a$ is also assumed to be probabilistic, that is, in addition to the plaintext and key, it also takes a random seed as input. A symmetric encryption scheme is a triple of algorithms $(\mathcal{K}_s, \mathcal{E}_s, \mathcal{D}_s)$ with probabilistic key generation $\mathcal{K}_s$, probabilistic encryption $\mathcal{E}_s$ and deterministic decryption $\mathcal{D}_s$; the key generation on input η outputs a single key, used both for encryption and decryption. We assume that the length of the ciphertext depends only on the length of the plaintext.

A.2. Computational model

A computational model is generated the following way: We consider all protocols for which the *roles* can be described by the language that we gave. In particular, the protocol may only contain generation of nonces, keys, symmetric and asymmetric encryptions, decryption, creation of pairs and taking them apart. We don't allow the honest participants to send their secret keys of public-key encryption, neither long term shared keys. Further, we assume that the encryptions are IND-CCA secure, and, in addition, the symmetric encryption is unforgeable. We also assume that long-term keys (public and shared) of honest parties are all generated honestly, as well as the honest parties' names. We assume that the length of encryptions and pairings depend only on the lengths of their inputs. All algorithms are assumed to be probabilistic polynomial time in the security parameter. An execution of a protocol in a malicious environment is defined by a set of honest PPT principals trying to execute the protocol, an unlimited number of possible other PPT principals acting arbitrarily, and a PPT adversary, who totally controls the communication between the parties (everything is sent through the adversary), and an initialization generating the long-term keys, fixing the names etc. The principals and the adversary can be represented as Turing machines, or, more conveniently, probabilistic random access machines. Unlimited number of parallel sessions are allowed. An execution is denoted by $\mathcal{X}$.

For each security parameter η, an execution determines a tree structure of the outcomes of the underlying subsequent coin tosses. We call this tree of coin tosses. The set of the branches of this tree, $\Omega^\eta_\mathcal{X}$, is the underlying probability field (with a sigma algebra structure that we don't detail here). The elements, ω are the branches of coin tosses, when all tosses are fixed. For a time t natural number (t, ω) determines the configuration of the execution $\xi(t, \omega)$ uniquely. Each pair of execution $\mathcal{X}$ together with a non-negligible sequence of subsets $\mathfrak{D} = (\mathfrak{D}^\eta)_{\eta\in\mathbb{N}}$ (with $\mathfrak{D}^\eta \subseteq \Omega^\eta_\mathcal{X}$) determines a model, $\mathcal{M}_{\mathcal{X},\mathfrak{D}}$.

The domains of interpretations of the different sorts are given the following way:

- `event`: $\Phi(\Delta)$ is a sequence of sets $\Phi(\Delta)^\eta \subseteq \mathfrak{D}^\eta$ and the probabilities of $\Phi(\Delta)^\eta$ give a non-negligible function in η.
- `timesection`: $\Phi(\tau)$ is a sequence of sections of the execution trees $\Phi(\tau)^\eta : \mathfrak{D}^\eta \to \mathbb{N}$ such that there is a polynomial time algorithm such that for any $t \in \mathbb{N}$ and $\omega \in \mathfrak{D}^\eta$, it can determine whether t equals $\Phi(\tau)^\eta(\omega)$ or not based on $\xi(1, \omega), ..., \xi(t, \omega)$.
- `bitstring`: $\Phi(m)$ is a sequences of random variables such that $\Phi(m)^\eta : \mathfrak{D}^\eta \to \{0, 1\}^* \cup \{\bot\}$.
- `bittree`: $\Phi(t)$ is a labeled ordered finite tree with an element of the interpretation domain of sort `bitstring` on each leaf and one of the function symbols $Pair$, $LPKEnc$, $LSKEnc$, $SKEnc$ on each of the internal nodes. The child nodes have to match the arities of the function symbols. Such an element looks exactly the same as the tree in Figure 1, except that the leafs are labelled with sequences of random variables.
- `hname` is the sort of the names of honest (uncorrupted) principals (we don't model dynamic corruption here).
- `name` represent principals in general. They have to be fixed before the run of the protocol, their public keys and long term shared keys also fixed in advance. Their keys do not have to be correctly generated. Any principal's name may vary with the security parameter.
- `hnonce` means honest nonces, that is, ones that are honestly generated by some principal, with the correct distribution and independently of everything that happened before. The distribution of nonces change with the security parameter, namely, become longer
- `hkey` means honest shared keys, that is, ones that are honestly generated by some principal (usually a trusted party), with the correct distribution and independently of everything that happened before. Clearly, they depend on the security parameter.
- `hseed` represent honestly generated random inputs of encryptions. Also depends on the security parameter.
- `sessionid` represents the session id's that keep track of the principals' sessions. May also depend on the security parameter.

Let $\Phi(T)$ denote the interpretation of a term. For any variable, t, $\Phi(t)$ is an element in one of the above domains, depending on the sort of t. Interpretation of function symbols and terms are defined the following way:

- $\Phi(\langle T_1, T_2\rangle)$ is the tree with root node labelled $Pair$ and its two ordered children are roots of the trees $\Phi(T_1)$ and $\Phi(T_2)$.
- $\Phi(\{T\}^s_Q)$ is defined similarly: It is the tree with the root node labelled as $PKEnc$, and its first chiled is labelled by $\Phi(Q)$, the second is the root of $\Phi(T)$, the third is labeled by $\Phi(s)$. $\Phi(\{T\}^s_{QQ'})$ and $\Phi(\{T\}^s_k)$ are defined similarly.
- $\Phi(\overline{T})$ is a sequence of random variables because $\overline{T}$ is of sort `bitstring`. It is computed the following way: Its value $\Phi(\overline{T})^\eta(\omega)$ is given by taking the values given by the labels on the leafs at η and ω, and carrying out the encryptions and pairings indicated by the labels on

the tree. For example,

$$\Phi(\overline{\{\langle N, K\rangle^s_Q})^\eta(\omega) = \\ \mathcal{E}_a([\mathcal{K}^\eta_Q(\omega), \Phi(N)^\eta(\omega), \Phi(K)^\eta(\omega)], \Phi(s)^\eta(\omega))$$

Where $\mathcal{K}_Q$ is the key of Q.

Finally, the interpretation of predicates:

- $t =_\Delta t'$ means that the tree structure of $\Phi(t)$ and $\Phi(t')$ are identical, with identical internal labels, and the random variables labeling the leafs that are not of sort `coin` are identical up to negligible probability, and further, if we compute the operations indicated by the tree, the results are also the same except for negligible probability on $\Phi(\Delta)$. $\Delta = \Delta'$ means $P((\Phi(\Delta)^\eta \cup \Phi(\Delta')^\eta) \setminus (\Phi(\Delta)^\eta \cap \Phi(\Delta')^\eta))$ is negligible in η, that is, $\Phi(\Delta)^\eta$ and $\Phi(\Delta)^\eta$ are the same up to negligible probability. $\tau =_\Delta \tau'$ means $P(\omega|\omega \in \Phi(\Delta)^\eta \wedge \Phi(\tau)^\eta(\omega) \neq \Phi(\tau')^\eta(\omega))$ is negligible in η.
- $t \sqsubseteq_\Delta t'$ means $\Phi(t)$ equals in the above sense on $\Phi(\Delta)$ to a subtree of $\Phi(t')$, such that it is not a name or seed argument of an encryption.
- $\tau <_\Delta \tau'$ means $P(\omega|\omega \in \Phi(\Delta)^\eta \wedge \Phi(\tau)^\eta(\omega) \not< \Phi(\tau')^\eta(\omega))$ is negligible in η.
- $\Delta \subseteq \Delta'$ means $P(\Phi(\Delta)^\eta \setminus \Phi(\Delta')^\eta)$ is negligible
- $Q\ generates^i_\tau N\big|_\Delta$
  Up to negligible probability, for $\omega \in \Phi(\Delta)^\eta$, $\xi(\tau(\omega), \omega)$ indicates the generation of a nonce by Q in its session i and writing it in a designated register.
- $Q\ receives^i_\tau t\big|_\Delta$
  Up to negligible probability, for $\omega \in \Phi(\Delta)^\eta$, $\xi(\tau(\omega), \omega)$ indicates that Q received a message which is than parsed the way t indicates it, each subterm being recorded in some register.
- $Q\ sends^i_\tau t\big|_\Delta$
  Up to negligible probability, for $\omega \in \Phi(\Delta)^\eta$, $\xi(\tau(\omega), \omega)$ indicates that Q sent a message which was previously put together the way t indicates it.
- $Sec_\tau(\langle A_1, ...A_n\rangle, t, \nu)\big|_\Delta$
  Satisfaction of this formula means the following: An adversary that is provided with $\Phi(t)^\eta(\omega)$ as well as the joint information available on ω until $\tau(\omega)^\eta$ to the principals other $A_1, ...A_n$ and to the protocol adversary, cannot distinguish ν from a nonce (or key respectively) generated independently of the protocol (with the same distribution that ν has restricted to $\Phi(\Delta)^\eta$.
- $KeySec_\tau(\langle A_1, ...A_n\rangle, t, K)\big|_\Delta$
  This is called *key-secrecy predicate*. It is satisfied if $A_1, ...A_n$ can safely encrypt messages using K until (not including) τ in Δ, even if t is given to them. By this we mean, that an adversary playing the security game for the encryption, cannot win the game even if provided with $\Phi(t)^\eta(\omega)$ as well as the joint information available on ω until $\tau(\omega)^\eta$ to the principals other $A_1, ...A_n$ and to the protocol adversary.

With these definitions, and if the encryptions and pairings satisfy the above mentioned security properties, the axioms that we listed are sound. A completely rigorous proof using probabilistic random access machines is being prepared for submission to a journal.

B. Possible variations

We would like to indicate some possible variations to the syntax that we gave.

B.1. Sorts

For example, when the lengths of keys and nonces are fixed, it make sense to have the following sort structure:

$$\left.\begin{array}{r} \texttt{hseed} \\ \texttt{hname} \subseteq \texttt{name} \\ \texttt{hnonce} \subseteq \texttt{nonce} \\ \texttt{hkey} \subseteq \texttt{key} \\ \texttt{sessionid} \end{array}\right\} \subseteq \texttt{bitstring} \subseteq \texttt{bittree}$$

Our analysis would still be valid without modifications, but the meaning of notation would be a little different. We actually did assume that lengths of nonces and keys are fixed,

`hseed`	$r, r', \ldots, r_1, r_2$
`hname`	$A, B, \ldots, A_1, A_2, \ldots$
`name`	$Q, Q' \ldots, Q_1, Q_2, \ldots$
`hnonce`	$N, N', \ldots, N_1, N_2, \ldots$
`nonce`	$n, n', \ldots, n_1, n_2, \ldots$
`hkey`	$K, K', \ldots, K_1, K_2, \ldots$
`key`	$k, k', \ldots, k_1, k_2, \ldots$
`nonce` or `hkey`	$\nu, \nu', \ldots, \nu_1, \nu_2, \ldots$
`sessionid`	$i, i', \ldots, i_1, i_2, \ldots$
`bitstring`	$s, s', \ldots, s_1, s_2, \ldots$ $m, m', \ldots, m_1, m_2, \ldots$
`bittree`	$t, t', \ldots, t_1, t_2, \ldots$ $u, u', \ldots, u_1, u_2, \ldots$
`timesection`	$\tau, \tau', \ldots, \tau_1, \tau_2, \ldots$
`event`	$\Delta, \Delta', \ldots, \Delta_1, \Delta_2, \ldots$
any variable	$v, v', \ldots, v_1, v_2, \ldots$

Table 2. Notation of variables

but we did not include `nonce` and `key` because we wanted axioms that work for stricter security (and variable nonce and key length) too, namely when length is hidden.

B.2. Reveal

As we mentioned, axiom VI.2 is not written to account for the reveal action. To incorporate that, we would need two properties:

$$
\begin{aligned}
&[Key]SecSend(\vec{A}, C, C', Reveal) \\
&\quad \wedge \ \big(C[\nu] \wedge C'[u] \wedge \nu \not\sqsubseteq u\big)\big|_{\Delta} \\
&\quad \rightarrow \ [Key]Sec_{\infty}(\vec{A}, u, \nu)\big|_{\Delta} \\
&\quad \vee \exists \Delta' \tau \vec{m}\big(\Delta' \subseteq \Delta \\
&\qquad \wedge ([Key]Sec_{\tau}(\vec{A}, u, \nu) \wedge Reveal_{\tau}(\vec{A}, \nu))|_{\Delta'}\big)
\end{aligned}
$$

and

$$
\begin{aligned}
&[Key]SecSend(\vec{A}, C, C', Reveal) \\
&\quad \wedge \ \big(C[\nu] \wedge C'[u] \wedge \nu \not\sqsubseteq u \wedge Reveal_{\tau}(\vec{A}, \nu)\big)\big|_{\Delta} \\
&\quad \rightarrow \ [Key]Sec_{\tau}(\vec{A}, u, \nu)\big|_{\Delta}
\end{aligned}
$$

B.3. Shared-key unforgeability

As we mentioned, it would also make sense to write the axioms for existential unforgeability. In that case, we have

(4) Authentication with long-term shared key:

$$
\begin{aligned}
&A\ receives^{i_2}_{\tau_2} t_2 \ \wedge \ \{t\}^s_{AB} \sqsubseteq t_2 \\
&\quad \rightarrow \exists A' i_1 t_1 t' ((A' = A \ \vee \ A' = B) \\
&\quad \wedge A'\ sends^{i_1}_{\tau_1} t_1; A\ receives^{i_2}_{\tau_2} t_2 \wedge \{t'\}^r_{AB} \sqsubseteq t_1 \\
&\quad \wedge \overline{\{t\}^s_{AB}} = \overline{\{t'\}^r_{AB}})
\end{aligned}
$$

(5) Authentication with shared session key

(a)

$$
\begin{aligned}
&KeySec_{\tau_2}(\vec{A}, K) \wedge A\ receives^{i_2}_{\tau_2} t_2 \ \wedge \ \{t\}^s_K \sqsubseteq t_2 \\
&\quad \rightarrow \exists A' i_1 t_1 t' r \tau_1 (A' \sqsubseteq \vec{A} \wedge A'\ sends^{i_1}_{\tau_1} t_1; A\ receives^{i_2}_{\tau_2} t_2 \\
&\quad \wedge \{t'\}^r_K \sqsubseteq t_1 \ \wedge \ \overline{\{t\}^s_K} = \overline{\{t'\}^r_K})
\end{aligned}
$$

With these properties, agreement and authentication from the initiator's proof in the sNS protocol can also be proven.