Technical Report

UCAM-CL-TR-460
ISSN 1476-2986

Number 460

Message reception in the inductive approach

Giampaolo Bella

March 1999

ISSN 1476-2986

Abstract

Cryptographic protocols can be formally analysed in great detail by means of Paulson's Inductive Approach, which is mechanised by the theorem prover Isabelle. The approach only relied on message sending (and noting) in order to keep the models simple. We introduce a new event, message reception, and show that the price paid in terms of runtime is negligible because old proofs can be reused. On the other hand, the new event enhances the global expressiveness, and makes it possible to define an accurate notion of agents' knowledge, which extends and replaces Paulson's notion of spy's knowledge. We have designed new guarantees to assure each agent that the peer does know the crucial message items of the session. The approach has now a broader scope. We provide general guidance to update the protocols analysed so far, and exemplify this on some of them.

1 Introduction

The "Inductive Approach" to analyse cryptographic protocols was developed by Paulson in 1997 [10]. Some inductive features had been formerly introduced in the field by Meadows [6], but Paulson's approach breaks the ground towards formal protocol analyses that merely rely on the expressiveness of induction.

The generic theorem prover Isabelle is used to mechanise the analysis. Vital properties such as confidentiality and authentication have been widely examined of several protocols (e.g. TLS [9], Kerberos [1, 2]), in the presence of a malicious agent who can actively tamper with the information retrieved from the traffic. However, the search for flaws can be performed more quickly by *state-enumeration* techniques on systems of limited size [4, 7, 11].

The Inductive Approach is based on the notion of event. All protocols tackled thus far can be analysed in terms of a single event, message sending, excepted the TLS protocol where agents also need to take note of some message items for future use. The number of events should be small in order to keep the model simple. On the other hand, many events could enrich the treatment. The best compromise seems modelling events that carry a high level of expressiveness.

This paper presents the introduction of a new, crucial event in the model: message reception. The enhanced model is (expectedly) more adherent to the real world, a feature that always improves readability. As a matter of fact, the guarantees for agents — protocol properties enforceable on assumptions that agents can verify — can be now expressed with smaller formal overhead.

The major outcome is allowing an effective formalisation of the knowledge of agents. The rudimentary version introduced originally [8] was soon simplified to express the knowledge of a single agent, the eavesdropper, optimising the approach to reason about confidentiality [10]. This represented the state of the art before our work. However, there are real-world scenarios in which pairs of agents communicating via a specific protocol need to agree on certain data (e.g. [3, 5]). A new hierarchy of properties based on agents' knowledge become crucial in this context, but their analyses require a broad formal notion of knowledge. This is achieved by our model.

The inductive definition of each protocol must be extended by one rule allowing reception of those messages that have been sent. However, the full proof scripts only pay a negligible price in terms of computational time. To achieve this, several general lemmas have been proved about message reception and agents knowledge (general facts only have to be proved once), which makes it possible to reuse the existing proofs.

Moreover, allowing for message reception is a mandatory step towards the analysis of non-repudiation protocols. Future work includes looking at properties such as non-repudiation of reception (e.g. [12]), for which we expect the present model to scale up easily.

Sec. 2 briefly gives few guidelines about the Inductive Approach. Our updates are described in sec. 3. The protocol analyses performed so far can be easily adapted to the new model, and gain expressiveness. This is exemplified together with the new *session key knowledge* and *nonce knowledge* theorems in sec. 4. Sec. 5 concludes the presentation.

2 Overview of the Inductive Approach

This section only introduces those features that are relevant to our treatment. The underlying intuition of the approach is rather simple. The real-world protocol must guarantee certain properties to hold during its sessions. Those properties can be verified as invariants of the formal protocol model, which specifies all admissible behaviours of an unbound population of agents, plus an eavesdropper (the "spy" below).

The following events may occur on the network

$$\mathsf{Says}\,A\,B\,X \qquad \text{and} \qquad \mathsf{Notes}\,A\,X$$

expressing respectively agent A sending message X to B, and agent A noting message X in her internal state.

The formal protocol model defines inductively all traces of events that are admissible according to the real-world protocol. The model also allows traces to be extended when agents accidentally lose some valuable information to the spy (the "oops" case). In this circumstance, a trace is extended by the event $\mathsf{Notes}\,\mathsf{Spy}\,msg$, where msg contains the lost information (typically a session key and the nonces that identify its session). See [10] for more.

3 Enhancing the Approach by Message Reception

We extend the Isabelle datatype defining the network events by the event modelling agent A's reception of message X:

$$\mathsf{Gets}\,A\,X$$

In the real world, a message can be received only if it has been previously sent (*reception invariant*). The formal protocol model can easily enforce this: a trace *can* be extended by the event $\mathsf{Gets}\,B\,X$ only when there exists an agent A such that the event $\mathsf{Says}\,A\,B\,X$ is on the trace. This will be stated by the Reception rule of the formal protocol specification. The extension can be performed for an unbound number of times, formalising the fact that agents may receive the same message more than once.

Moreover, the model does not guarantee that messages that are sent will be ever received; they could even be received in a wrong order or by wrong recipients.

The oops case could be modelled by a Gets event. For example, consider the scenario in which the trusted server has sent an encrypted message containing a session key K, and an agent P has associated K to a nonce Np. Allow the spy to receive the message $\{\!|K, Np|\!\}$ if an oops occurs. However, this would compromise the reception invariant and consequently burden the proof script with several case splits. Using the Notes event avoids the problem.

The function used extracts from a trace all components of messages appearing on it [10]. Therefore, if e.g. a key K is fresh on a trace *evs*, we have $k \notin \mathsf{used}\ evs$. If a message is received on a trace, the set of components used on the trace is unaltered thanks to the reception invariant (since messages that are received have been previously sent, their components belong to the set used *evs* already). We extend the definition of used accordingly:

$$\mathsf{used}((\mathsf{Gets}\, A\, X)\ \#\ evs) \triangleq \mathsf{used}\ evs$$

3.1 Formalising the Agents' Knowledge

We introduce the function knows yielding the set of messages that an agent can handle on a given trace. It generalises (and replaces) Paulson's definition of the function spies [10] to any agent, and to the message reception event.

The function is defined inductively on the length of the trace. The base case states that agents know their respective initial state. Recall that the initial state of the trusted server is the set of all agents' long-term keys; the initial state of the spy is the set of all compromised agents' long-term keys (the spy is compromised too); the initial state of any other agent is the agent's long-term key.

$$\mathsf{knows}\, A\, [] \quad \triangleq \quad \mathsf{initState}\, A$$

Each agent knows what the agent alone sent on a trace. The spy knows all messages ever sent.

$$\mathsf{knows}\, A\, (\mathsf{Says}\, A'\, B X\ \#\ evs) \quad \triangleq \quad \begin{cases} \{X\} \cup \mathsf{knows}\, A\ evs & \text{if } A = A' \ \vee\ A = \mathsf{Spy} \\ \mathsf{knows}\, A\ evs & \text{otherwise} \end{cases}$$

Each agent knows what the agent alone noted on a trace. The spy also knows compromised agents' notes.

$$\mathsf{knows}\, A\, (\mathsf{Notes}\, A'\, X\ \#\ evs) \quad \triangleq \quad \begin{cases} \{X\} \cup \mathsf{knows}\, A\ evs & \text{if } A = A' \ \vee \\ & \quad (A = \mathsf{Spy}\ \wedge\ A' \in \mathsf{bad}) \\ \mathsf{knows}\, A\ evs & \text{otherwise} \end{cases}$$

Each agent knows what the agent alone received from a trace. Since messages that are received must have been previously sent, the spy knows them already. Therefore, the spy's knowledge does not grow on received messages.

$$\mathsf{knows}\, A\, (\mathsf{Gets}\, A'\, X\ \#\ evs) \quad \triangleq \quad \begin{cases} \{X\} \cup \mathsf{knows}\, A\ evs & \text{if } A = A' \\ \mathsf{knows}\, A\ evs & \text{otherwise} \end{cases}$$

In particular, the last case exploits the message reception event to strengthen Paulson's original definition of agents' knowledge: agent A could *see* message X when X had been sent by somebody to A ([8], sec. 4.5 "Events and Agent Knowledge"). This could not assure that A ever got hold of X and therefore knew it.

Recall that the function analz applies to a set of messages and extracts all components of compound messages and bodies of messages encrypted under keys that are recursively known. If in the real world an agent A knows a message X, the formal protocol model contains a trace *evs* such that either $X \in (\mathsf{knows}\, A\, evs)$ if A did not need any cryptanalysis to get hold of X, or $X \in \mathsf{analz}(\mathsf{knows}\, A\, evs)$ if A had to retrieve X from within a message of the set knows A *evs*.

3.2 Updating the Existing Specifications

The existing formal protocol specifications can be easily updated by the following procedure.

- Update the Fake rule replacing "spies" by "knows Spy".
- Update each rule replacing any event $\mathsf{Says}\, P\, Q\, X$, P being a free variable of the rule, by the event $\mathsf{Gets}\, Q\, X$. (This typically modifies the rule premises).
- Add the Reception rule.

In the original specification, an agent A could send a new message when some other (undefined) agent had sent A a suitable message X. Obviously, A could only check on reception of X whether that event had happened, because she could not monitor any events performed by other agents.

The message reception event can make the formal protocol specification closer to the real world. The new specification tells each agent which message the agent can send if the agent has previously sent and/or received certain messages. These conditions model directly the checks performed by a real-world agent before sending a new message. Therefore, the specification now provides a better basis for implementation. This is exemplified in sec. 4.1 and in Appendix.

3.3 Updating the Existing Theorems

The existing proof scripts can be trivially updated replacing the string "spies" by the string "knows Spy". "Session key secrecy" theorems [10] now state that the key K is such that $K \notin \mathsf{analz}(\mathsf{knows}\, \mathsf{Spy}\, evs)$ in certain circumstances.

Applying the reception invariant, we can prove that the spy knows all messages that are received by any agents. More formally, for any trace *evs* containing the event $\mathsf{Gets}\, A\, X$, we can state that

$$X \in \mathsf{knows\,Spy}\; evs$$

and, by $H \subseteq \mathsf{parts}\, H$ for any set H [10], that (*GetsSpyKnows*):

$$X \in \mathsf{parts}(\mathsf{knows\,Spy}\; evs)$$

Recall that the function parts applies to a set of messages and extracts all message components excepted the ciphers' encryption keys [10].

Consider a theorem that assumes $X \in \mathsf{parts}(\mathsf{knows\,Spy}\; evs)$. Let agent A be the intended recipient for message X, and suppose that the theorem assesses a result useful to A. Replace the mentioned assumption by the event $\mathsf{Gets}\, A\, X\, evs$. The resulting theorem can be proved applying lemma *GetsSpyKnows* and then the proof for the old theorem. The new theorem is a guarantee for A, as it rests on assumptions that A can verify. All theorems proved thus far can be so updated (see sec. 4.1).

3.4 Proving the Session Key Knowledge Theorem

The operations formalised by the function analz must also be performed by agents different from the spy who need to access the components of the messages they know. For instance, at latest at the end of a protocol session, the peers have to extract the session key they have agreed to use for their communication.

We have designed a new guarantee for A establishing that a key K is such that $K \in \mathsf{analz}(\mathsf{knows}\, B\; evs)$ on assumptions that A can verify (typically suitable message receptions). The theorem, called *session key knowledge* theorem (*for A about B over K*), assures A that B can analyse (extract) the key K from the messages B knows. A guarantee of the same form can be stated for B. The peers can so understand whether they know the same key (see sec. 4.2).

Normally, it is the trusted server who issues the session key that is sent to the peers. However, it is not guaranteed a priori that the peers will receive it. Therefore, a session key knowledge theorem for A about B over K can be achieved by the following strategy.

1. Prove that if A receives a suitable message on a trace, then B has received (and can decrypt) on the same trace another message containing K.

2. Apply a lemma stating that agents know the messages they received (thanks to the reception invariant).

3. Apply $H \subseteq \mathsf{analz}$ for any set H, and extraction of message components under analz.

Step 1 is proved using standard inductive techniques [10], and may require few intermediate steps to establish which messages must be sent before B receives his message. The lemma mentioned by step 2 is trivial.

This strategy may also allow proving agents' knowledge of other valuable data (e.g. nonces), by means of other lemmas stating that agents know what they sent and what they noted on a trace (see sec. 4.3).

4 The Outcomes

This section exemplifies the notions presented above, showing the outcomes in terms of improved expressiveness of existing analyses, and of new proofs about agents' knowledge.

4.1 Enhancing Expressiveness

We update the analysis of the Otway-Rees protocol performed by Paulson [10].

The new formal protocol specification about the version that encrypts the nonce Nb is presented in Appendix. According to the procedure presented in sec. 3.2, rule Fake gets a minor change, and rules OR2, OR3, OR4 gain new premises. The others remain unaltered. The specification is enriched by the Reception rule enforcing the reception invariant.

We have updated Paulson's theorems as described in sec. 3.3. For example, we can prove that if the events

$$\mathsf{Gets\ Server}\ \{\!|Na, \mathsf{Agent}\,A, \mathsf{Agent}\,B, \mathsf{Crypt}(\mathsf{shrK}\,A)\{\!|Na, \mathsf{Agent}\,A, \mathsf{Agent}\,B|\!\}, X|\!\}$$
$$\mathsf{Gets\ Server}\ \{\!|Na, \mathsf{Agent}\,A, \mathsf{Agent}\,B', \mathsf{Crypt}(\mathsf{shrK}\,A)\{\!|Na, \mathsf{Agent}\,A, \mathsf{Agent}\,B'|\!\}, X|\!\}$$

appear on a trace *evs*, then follows $B = B'$, provided that A is uncompromised. Thanks to the new form of OR3, this result can replace the original theorem expressed in terms of parts. However, its importance is mainly technical for other proofs.

The authenticity theorems now state more explicitly in what circumstances agents can consider their certificates to be authentic. If A has started the protocol with B, she has to check that two events concerning only herself

$$\mathsf{Gets}\ A\ \{\!|Na, \mathsf{Crypt}(\mathsf{shrK}\,A)\{\!|Na, \mathsf{Key}\,K|\!\}|\!\}$$
$$\mathsf{Says}\ A\ B\ \{\!|Na, \mathsf{Agent}\,A, \mathsf{Agent}\,B,$$
$$\mathsf{Crypt}(\mathsf{shrK}\,A)\{\!|Na, \mathsf{Agent}\,A, \mathsf{Agent}\,B|\!\}|\!\}$$

occurred on the trace, to infer that her certificate $\{\!|Na, K|\!\}_{Ka}$ originated with the server for some nonce Nb, i.e. the event

$$\mathsf{Says\ Server}\ B\ \{\!|Na, \mathsf{Crypt}(\mathsf{shrK}\,A)\{\!|Na, \mathsf{Key}\,K|\!\},$$
$$\mathsf{Crypt}(\mathsf{shrK}\,B)\{\!|Nb, \mathsf{Key}\,K|\!\}|\!\}$$

occurred on the same trace. At the other end, B can come to the same conclusion if the two events concerning himself only

$$\begin{aligned}&\mathsf{Gets}\,B\,\{\!|Na, X, \mathsf{Crypt}(\mathsf{shrK}\,B)\{\!|Nb, \mathsf{Key}\,K|\!\}|\!\}\\ &\mathsf{Says}\,B\ \mathsf{Server}\,\{\!|Na, \mathsf{Agent}\,A, \mathsf{Agent}\,B, X',\\ &\qquad\qquad \mathsf{Crypt}(\mathsf{shrK}\,B)\{\!|Na, Nb, \mathsf{Agent}\,A, \mathsf{Agent}\,B|\!\}|\!\}\end{aligned}$$

occurred on the trace; B is thus assured that his certificate $\{\!|Na, Nb, A, B|\!\}_{Kb}$ is authentic.

4.2 Proving Agents' Knowledge for a Shared-Key Protocol

We have updated Paulson's analysis of the shared-key Needham-Schroeder protocol [13] and proved the session key knowledge theorems and a nonce knowledge theorem about the protocol.

Suppose that A initiates a session of the protocol with B and that both agents are uncompromised. The session key knowledge theorem for A about B over K states that if the events

$$\begin{aligned}&\mathsf{Gets}\,A\,(\mathsf{Crypt}(\mathsf{shrK}\,A)\{\!|Na, \mathsf{Agent}\,B, \mathsf{Key}\,K, X|\!\})\\ &\mathsf{Gets}\,A\,(\mathsf{Crypt}\,K(\mathsf{Nonce}\,Nb))\end{aligned}$$

occur on a trace *evs*, and the key K has not been leaked by accident on the trace, then

$$\mathsf{Key}\,K \in \mathsf{analz}(\mathsf{knows}\,B\;evs)$$

Learning that B can access the key K is an important conclusion for A: it means that B will be able to understand the messages she encrypts under K. However, A must trust K to have not been lost accidentally to the spy. This requirement comes from the application of the session key secrecy theorem, and is therefore minimal and in common to several guarantees (e.g. [2]).

The proof applies the guarantee of authentication of B to A (the updated version of "A_trusts_NS4" [13]) to infer that the event

$$\mathsf{Says}\,B\,A\,(\mathsf{Crypt}\,K(\mathsf{Nonce}\,Nb))$$

occurs on *evs*. It is this result to require the session key secrecy theorem: if K were not confidential, then the spy could have forged $\{\!|Nb|\!\}_K$. At this stage, we prove by standard inductive techniques that also the event

$$\mathsf{Gets}\,B\,(\mathsf{Crypt}(\mathsf{shrK}\,B)\{\!|\mathsf{Key}\,K, \mathsf{Agent}\,A|\!\})$$

must occur on *evs*. Then, the proof applies the lemma stating that agents know the messages they receive, and finally the basic properties of analz.

On the same assumptions as those of the session key knowledge theorem for A, we can also prove a nonce knowledge theorem for A about B over Nb, concluding that

$$\mathsf{Nonce}\,Nb \in \mathsf{analz}(\mathsf{knows}\,B\ evs)$$

At the other end of the communication, B gets a symmetric guarantee. If the two events

$$\mathsf{Gets}\,B\ (\mathsf{Crypt}(\mathsf{shrK}\,B)\{\!|\mathsf{Key}\,K, \mathsf{Agent}\,A|\!\})$$
$$\mathsf{Gets}\,B\ (\mathsf{Crypt}\,K\{\!|\mathsf{Nonce}\,Nb, \mathsf{Nonce}\,N|\!\})$$

occur on a trace *evs*, and the key K has not been leaked by accident on the trace, then

$$\mathsf{Key}\,K \in \mathsf{analz}(\mathsf{knows}\,A\ evs)$$

Agent B so learns that A agrees on the same session key he is using.

We prove this by the same strategy applied to the guarantee for A. The first step applies the authentication guarantee of A to B (the updated version of "B_trusts_NS5" [13], which rests on one assumption fewer), to obtain that the event formalising the fifth step of the protocol

$$\mathsf{Says}\,A\,B\ (\mathsf{Crypt}\,K\{\!|\mathsf{Nonce}\,Nb, \mathsf{Nonce}\,Nb|\!\})$$

occurs on *evs*. The next step applies to this event a new lemma concluding that

$$\mathsf{Gets}\,A\ (\mathsf{Crypt}(\mathsf{shrK}\,A)\{\!|\mathsf{Nonce}\,Na, \mathsf{Agent}\,B, \mathsf{Key}\,K, X|\!\})$$

must occur on *evs* for some nonce Na. The final step is straightforward. The full proof script, inclusive of the authentication guarantee, executes in approximately 10 seconds on a Pentium Pro 300MhZ, the same runtime of the guarantee for A.

4.3 Proving Agents' Knowledge for a Public-Key Protocol

We have updated the existing analysis of the public-key Needham-Schroeder protocol [10].

The protocol merely exchanges nonces in order to authenticate the peers to each other. Looking at the (well-known) protocol messages, it is relatively easy to realise that A at some point gets evidence that B knew her nonce Na and was therefore present on the network. Spotting a similar guarantee for B requires analogous effort. However, such properties about knowledge can be now proved formally.

Suppose that A runs the protocol with B and that they are uncompromised. Let *evs* be a trace of the protocol. If *evs* contains the events

$$\begin{aligned}&\mathsf{Says}\,A\,B\,(\mathsf{Crypt}(\mathsf{pubK}\,B)\{\!|\mathsf{Nonce}\,Na, \mathsf{Agent}\,A|\!\})\\ &\mathsf{Gets}\,A\,(\mathsf{Crypt}(\mathsf{pubK}\,A)\{\!|\mathsf{Nonce}\,Na, \mathsf{Nonce}\,Nb|\!\})\end{aligned}$$

A can invoke the guarantee that authenticates B to her (the updated version of that in [10], sec. 5.2) learning that *evs* must contain the event

$$\mathsf{Says}\,B\,A\,(\mathsf{Crypt}(\mathsf{pubK}\,A)\{\!|\mathsf{Nonce}\,Na, \mathsf{Nonce}\,Nb|\!\}) \tag{1}$$

On this assumption, we have proved by standard inductive techniques that also the event

$$\mathsf{Gets}\,B\,(\mathsf{Crypt}(\mathsf{pubK}\,B)\{\!|\mathsf{Nonce}\,Na, \mathsf{Agent}\,A|\!\}) \tag{2}$$

must occur on *evs*, and therefore, since $\mathsf{priK}\,B \in \mathsf{analz}(\mathsf{knows}\,B\,evs)$, follows

$$\mathsf{Nonce}\,Na \in \mathsf{analz}(\mathsf{knows}\,B\,evs)$$

This guarantee is the *nonce knowledge* theorem *for A about B over Na*. It provides A with simple assumptions to check in order to learn that B knows her nonce.

The nonce knowledge theorem for B about A over Nb is analogous. If the events

$$\begin{aligned}&\mathsf{Says}\,B\,A\,(\mathsf{Crypt}(\mathsf{pubK}\,A)\{\!|\mathsf{Nonce}\,Na, \mathsf{Nonce}\,Nb|\!\})\\ &\mathsf{Gets}\,B\,(\mathsf{Crypt}(\mathsf{pubK}\,B)(\mathsf{Nonce}\,Nb))\end{aligned}$$

occur on *evs*, then, by authentication of A to B (the updated version of that in [10], sec. 5.4), the trace must contain the event

$$\mathsf{Says}\,A\,B\,(\mathsf{Crypt}(\mathsf{pubK}\,B)(\mathsf{Nonce}\,Nb)) \tag{3}$$

A proof by induction leads to the fact that the trace must also contain the event

$$\mathsf{Gets}\,A\,(\mathsf{Crypt}(\mathsf{pubK}\,A)\{\!|\mathsf{Nonce}\,Na, \mathsf{Nonce}\,Nb|\!\}) \tag{4}$$

for some nonce Na, and therefore

$$\mathsf{Nonce}\,Nb \in \mathsf{analz}(\mathsf{knows}\,A\,evs)$$

Our proofs were not affected by Lowe's middle-person attack [4]. Therefore, if an agent A starts a session with the spy, another agent B may on another session issue a nonce Nb for talking to A and learn that A does know Nb, although B is in fact talking to the spy. This supports the claim that the mere knowledge of the nonce does not guarantee authentication. Obviously, the same guarantees have been proved for the Needham-Schroeder-Lowe protocol.

4.4 More on our Formalisation of Knowledge

The nonce knowledge theorems proved for the public-key Needham-Schroeder protocol stimulates some further discussion about our formalisation of knowledge.

To achieve the guarantees, we had to prove that event (1) implied event (2), and that event (3) implied event (4). However, these steps seem redundant because B knows Na already when he creates the message $\{\!|Na, Nb|\!\}_{Ka}$ in event (1); similarly, A knows Nb already when she creates the message $\{\!|Nb|\!\}_{Kb}$ in event (3).

Those steps are in fact necessary, because in our model an agent knows a message item if and only if the agent is able to access it from one of the messages that he/she either sends or notes or receives. With shared-key protocols, this definition also captures those items that the agent may invent and send on the network for their first time, since the agent can send messages either in cleartext or encrypted under keys that he/she knows. This is not the case with public-key protocols. An agent may encrypt some items using the public key of the peer, and send the cipher: at this stage, those items are no longer accessible to the agent, and therefore not captured by our formalisation of knowledge.

A trivial attempt to solve this problem extends the definition of knows by

$$\mathsf{knows}\,A\,(\mathsf{Says}\,A'\,B(\mathsf{Crypt}(\mathsf{pubK}\,B)X)\ \#\ evs) \triangleq \begin{cases} \{X\} \cup \mathsf{knows}\,A\,evs & \text{if } A = A' \\ \mathsf{knows}\,A\,evs & \text{otherwise} \end{cases}$$

so to capture the knowledge of those items that are sent encrypted under the public key of the recipient. This definition would work well with Needham-Schroeder, and the proofs would speed up. However, it would fail on a protocol that required a third agent C to create $\{\!|X|\!\}_{Kb}$, and then A to forward it to B. In this case, A should not be allowed to access X because she does not know Kb^{-1}.

The problem is solved as follows. If the items sent inside the cipher have been received, they are known anyway, as exploited by our proofs. On the contrary, if they are invented at the moment, the agent should note them (by a Notes event), as suggested by Paulson in his analysis of the TLS protocol [9], and our definition of knowledge would capture them.

5 Conclusions

Cryptographic protocols can be deeply analysed by means of Paulson's Inductive Approach in Higher Order Logic. Their crucial properties can be established by theorems proved with the support of the theorem prover Isabelle.

The approach intentionally avoided message reception in order to simplify the mechanisation. We have added a reception event, so that the protocol traces also record information about which messages have been received, which agents have received them, and at which stage of the session they were received. We have proved some basic lemmas that allow the old proofs to be straightforwardly reused. Consequently, the existing models can be trivially updated, and the runtime is negligibly increased.

The work initially aimed at improving the analyses in terms of expressiveness. This aim has been achieved, as the guarantees for each agent can now be expressed upon events that concern the agent alone, and not on conditions about the network.

The new model turned out to provide a good basis for a broad definition of agents' knowledge. This is a new feature for the approach, since Paulson's initial attempt [8] had been soon tailored to a formalisation of the spy's knowledge only [10]. We allow each agent but the spy to *know* only the messages the agent sends, notes, or receives, while the spy *knows* all messages ever sent, and those noted by compromised agents. The knowledge of a message item is formalised by the ability to access it (possibly decrypting some ciphers) from one of the messages that are *known*.

We have designed a new hierarchy of guarantees based on agents' knowledge: the *session key knowledge* theorems for shared-key protocols, and the *nonce knowledge* theorems, which can also be proved for public-key protocols. They assure each agent about the peer's knowledge of a session key or of a nonce.

The potentialities of the Inductive Approach appear significantly enhanced, also towards the analysis of non-repudiation protocols. Our model will be publicly available with the next Isabelle distribution.

Acknowledgements. Larry Paulson commented extensively on a draft of this paper. The author wishes to dedicate this work to Maria Matarazzo.

References

[1] G. Bella, L. C. Paulson. Mechanising BAN Kerberos by the Inductive Method. Proc. of *Tenth Conference on Computer Aided Verification*, Springer, LNCS 1427, 1998.

[2] G. Bella, L. C. Paulson. Kerberos Version IV: Inductive Analysis of the Secrecy Goals. Proc. of *Fifth European Symposium on Research in Computer Security*, Springer, LNCS 1485, 1998.

[3] M. Burrows, M. Abadi, R. M. Needham. A logic of authentication. *Proceedings of the Royal Society of London*, 426:233-271, 1989.

[4] G. Lowe. Breaking and Fixing the Needham-Schroeder Public-Key Protocol using FDR. In *Tools and Algorithms for the Construction and*

Analysis of Systems, Margaria and Steffen (eds.), LNCS1055, Springer Verlag, 147-166, 1996.

[5] G. Lowe. A Hierarchy of Authentication Specifications. Proc. of *Tenth IEEE Computer Security Foundations Workshop*, 1997.

[6] C. Meadows. The NRL Protocol Analyzer: An Overview. *Journal of Logic Programming*, 26(2), 113-131, 1996.

[7] J. C. Mitchell, M. Mitchell, U. Stern: Automated Analysis of Cryptographic Protocols Using Murphi. In *Proc. of Symposium on Security and Privacy*, IEEE Press, 1997.

[8] L. C. Paulson. Proving properties of security protocols by induction. 10th Computer Security Foundations Workshop (June 1997), 70-83.

[9] L. C. Paulson. Inductive Analysis of the Internet Protocol TLS. Cambridge University, Computer Laboratory, *Technical Report No. 440*, July 1997.

[10] L. C. Paulson. The Inductive Approach to Verifying Cryptographic Protocols. *Journal of Computer Security*, 6:85-128, 1998.

[11] S. Schneider. Security Properties and CSP. In *Proc. of Symposium on Security and Privacy*, IEEE Press, 1996.

[12] J. Zhou, D. Gollmann. A fair Non-Repudiation Protocol. In *Proc of Symposium on Security and Privacy*, IEEE Press, 1996.

[13] See theory "NS_Shared" at `http://www4.informatik.tu-muenchen.de/~isabelle/library/HOL/Auth/`

A Exploiting message reception to specify Otway-Rees

```
Nil  [] ∈ otway

Fake [| evs ∈ otway;  X ∈ synth (analz (knows Spy evs)) |]
     ⟹ Says Spy B X  # evs ∈ otway

OR1  [| evs1 ∈ otway; Nonce NA ∉ used evs1 |]
     ⟹ Says A B {|Nonce NA, Agent A, Agent B,
                  Crypt (shrK A) {|Nonce NA, Agent A, Agent B|}|}
        # evs1 ∈ otway

OR2  [| evs2 ∈ otway; Nonce NB ∉ used evs2;
        Gets B {|Nonce NA, Agent A, Agent B, X|} ∈ set evs2 |]
     ⟹ Says B Server
                 {|Nonce NA, Agent A, Agent B, X,
                   Crypt (shrK B)
                     {|Nonce NA, Nonce NB, Agent A, Agent B|}|}
        # evs2 ∈ otway

OR3  [| evs3 ∈ otway; Key KAB ∉ used evs3;
        Gets Server
             {|Nonce NA, Agent A, Agent B,
               Crypt (shrK A) {|Nonce NA, Agent A, Agent B|},
               Crypt (shrK B) {|Nonce NA, Nonce NB, Agent A, Agent B|}|}
          ∈ set evs3 |]
     ⟹ Says Server B
             {|Nonce NA,
               Crypt (shrK A) {|Nonce NA, Key KAB|},
               Crypt (shrK B) {|Nonce NB, Key KAB|}|}
          # evs3 ∈ otway

OR4  [| evs4 ∈ otway; B ≠ Server;
        Says B Server {|Nonce NA, Agent A, Agent B, X',
                        Crypt (shrK B)
                          {|Nonce NA, Nonce NB, Agent A, Agent B|}|}
          ∈ set evs4;
        Gets B {|Nonce NA, X, Crypt (shrK B) {|Nonce NB, Key K|}|}
          ∈ set evs4 |]
     ⟹ Says B A {|Nonce NA, X|} # evs4 ∈ otway

Oops [| evso∈ otway;
        Says Server B {|Nonce NA, X,
                        Crypt (shrK B) {|Nonce NB, Key K|}|}
          ∈ set evso |]
     ⟹ Notes Spy {|Nonce NA, Nonce NB, Key K|} # evso ∈ otway

Reception [| evsr ∈ otway; Says A B X ∈ set evsr |]
          ⟹ Gets B X # evsr ∈ otway
```