Modelling Agents' Knowledge Inductively

Giampaolo Bella

Computer Laboratory — University of Cambridge
New Museums Site, Pembroke Street
Cambridge CB2 3QG (UK)

Giampaolo.Bella@cl.cam.ac.uk

Abstract. This note introduces my extensions to Paulson's "Inductive Approach" in order to reason about agents' knowledge. First, I argue that an agent knows all the components of those messages she creates. I exploit this notion on some crypto-protocols with particular message structure. Then, I allow a new event, message reception, to occur on the network. This leads to a broad definition of agents' knowledge, which extends the existing definition of spy's knowledge. I will discuss strengths and weaknesses of the two models when proving session key knowledge.

1 Overview

Paulson's "Inductive Approach" [6] is a powerful framework to reason about the confidentiality goals [4] met by modern cryptographic protocols. The expressiveness of the inductive definitions provides realistic models (agents may conspire with the spy, session keys may be accidentally lost, etc.), while the theorem prover Isabelle supports the mechanisations of the proofs.

The approach relies on the concept of *trace*, which is a list of events that can occur on the network. The protocol model is defined by specifying how to build inductively all possible traces of events, according to the message exchanges required by the real-world protocol. Specifying all possible traces signifies that the model includes all possible situations arising when an infinite population of agents run the protocol. The spy is amongst these agents.

The first considerations about agents' knowledge in the Inductive Approach arose when I formulated the theorems stating confidentiality of session keys in such a way that the agents could verify their assumptions [3]. By invoking these theorems, the agents are able to *learn* that certain session keys are confidential, so increasing their knowledge.

Other approaches allow deeper reasoning about knowledge. For instance, the BAN logic — though criticised about its soundness — contains a predicate of the form "A and B believe that they may use K to communicate", while state-enumeration techniques can enforce the peers' agreement on a set of data [5]. Both approaches allow reasoning about the agents' knowledge of relevant message items.

So, I started to investigate the matter within the Inductive Approach, focusing on the knowledge of session keys. A pair of peers A and B typically consider

a key K "good" for their communication when they both have evidence that K is confidential, and that K is *known* to both of them. I will concentrate on the latter requirement.

As a general remark, I must point out that crypto-protocols are conventionally run by a population of agents (also said network parties, or principals), when in fact they are run by the agents' workstations. This difference is rarely pointed out in the literature (e.g. [1]). I will hold to the convention and speak about agents' knowledge instead of workstations' memory cells. I will not deal with the authentication of agents to workstations.

2 Model 1: hacking with message creation

Some of the shared-key protocols analysed inductively so far have a common feature. After the trusted server has created a new session key and the peers have obtained it, each agent uses the session key to encrypt a *new* message, and then sends it to the peer. Although this design could be criticised as an incautious way to achieve authentication, it can be exploited to prove that each agent knows the session key.

One possible strategy to prove this is showing that the agent is the true creator of the message encrypted under the session key. The ability to use a key to build a cipher obviously implies the knowledge of the key.

More formally, an agent A is the creator of a message msg meant for her peer B on some trace evs, when A has sent some message msg', containing msg as a component, to B on evs, and msg never appeared before this event:

$$
\begin{array}{l}
A \ \mathsf{Issues}\ B\ \mathsf{with}\ msg\ \mathsf{on}\ evs \equiv \\
\quad \exists\, msg'.\ \mathsf{Says}\, A\, B\ msg' \in \mathsf{set}\ evs\ \wedge\ msg \in \mathsf{parts}\{msg'\}\ \wedge \\
\quad msg \notin \mathsf{parts}(\mathsf{spies}\,(\mathsf{takeWhile}(\lambda x.\, x \neq \mathsf{Says}\, A\, B\ msg')(\mathsf{rev}\ evs)))
\end{array}
$$

Recall that $\mathsf{parts}(\mathsf{spies}\ evs)$ yields all message components of all messages in the trace evs, assuming readable ciphers. Traces are always extended from the head, therefore the "rev" function must be applied.

$$
\begin{array}{l}
1.\ A \to \mathsf{S} : A, B, Na \\
2.\ \mathsf{S} \to A : \{\!|Na, B, Kab, \underbrace{\{\!|Kab, A|\!\}_{Kb}}_{ticket}|\!\}_{Ka} \\
3.\ A \to B : \underbrace{\{\!|Kab, A|\!\}_{Kb}}_{ticket} \\
4.\ B \to A : \{\!|Nb|\!\}_{Kab} \\
5.\ A \to B : \{\!|Nb - 1|\!\}_{Kab}
\end{array}
$$

Fig. 1. The shared-key Needham-Schroeder protocol

I have proved that the Issues predicate holds on several messages of BAN Kerberos, Kerberos IV, and the shared-key Needham-Schroeder protocol (Fig. 1).

For example, B's knowledge of the session key on the Needham-Schroeder protocol (**Lemma 1**) can be stated as follows: if B sends message 4 of the protocol, i.e. the event

$$\mathsf{Says}\, B\, A\, (\mathsf{Crypt}\, K\, (\mathsf{Nonce}\, Nb)) \tag{1}$$

appears on a trace *evs*, then

$$B\ \mathsf{Issues}\ A\ \mathsf{with}\ (\mathsf{Crypt}\, K\, (\mathsf{Nonce}\, Nb))\ \mathsf{on}\ evs$$

The conclusion assesses that B is the true creator of the cipher, which can be read as B *knows* K. Lemmas of this form usually require a confidentiality assumption on the key K, otherwise the spy could forge the cipher and confute the conclusion. However, in this case B encrypts a fresh nonce so the spy cannot tamper with it.

Clearly, this technique cannot be applied if the peers do not use the session key after its reception. However, the peers do know the session key *when* they receive it. To express this property, I decided to model message reception.

3 Model 2: defining agents' knowledge

The event $\mathsf{Gets}\, A\, msg$ formalises agent A receiving message *msg*. The protocol model allows a message sent by A to B to be possibly (not necessarily for the network is insecure) received by B. The same message can be received more than once, and even by the wrong recipient [2]. Incidentally, this model must be used to formalise non-repudiation protocols and express properties like non-repudiation of reception.

At this stage, I have defined a function knows that takes as parameters an agent and a trace, yielding all messages that the agent has handled on the trace. The base case states that an agent knows her initial state. The inductive steps (the case for the Notes event is omitted here for brevity) state that each agent knows what she sends or receives, while the spy knows all messages ever sent.

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

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

Recall that a message can be received only if it was sent, therefore the inductive step on Gets does not need to enrich the spy's knowledge. This definition extends and replaces the existing definition of the function spies [6].

Agent A's knowledge of a message *msg* is formalised by $msg \in (\mathsf{knows}\, A\, evs)$ for some trace *evs*, while A's knowledge of a message component, say a key K, is expressed as $\mathsf{Key}\, K \in \mathsf{analz}(\mathsf{knows}\, A\, evs)$, meaning that A must extract K from a message amongst those that she knows.

In this model, B's knowledge of the session key on Needham-Schroeder (**Lemma 2**) states that if B receives the "ticket" sealed under his shared key, i.e. the event

$$\mathsf{Gets}\, B\ (\mathsf{Crypt}(\mathsf{shrK}\, B)\, \{|\mathsf{Key}\, K, \mathsf{Agent}\, A|\}) \tag{2}$$

appears on a trace *evs*, then B can access K from the traffic she has handled,

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

4 Proving more significant guarantees

At this stage, I want to make the above results available to the respective peers. More concretely, while those lemmas merely tell the agents what themselves know, I will now build up from them more crucial guarantees telling the agents what their peers know. I will still concentrate on knowledge of session keys.

In Model 1, I can easily prove by induction that if

$$\begin{aligned}&\mathsf{Crypt}\ (\mathsf{shrK}\, A)\, \{|N, \mathsf{Agent}\, B, \mathsf{Key}\, K, X|\} \in \mathsf{parts}(\mathsf{spies}\, evs)\\ &\mathsf{Crypt}\ K\ (\mathsf{Nonce}\, Nb) \in \mathsf{parts}(\mathsf{spies}\, evs)\end{aligned}$$

and K is confidential, then event (1) must be on the trace *evs*. The message $m = \{|N, B, K, X|\}_{Ka}$ is required to appear in the traffic on *evs* to pinpoint the peers for K. Confidentiality over K is needed to prevent the spy from forging the message $m' = \{|Nb|\}_K$ (not needed in Lemma 1 because Nb was assumed to be the fresh nonce created in step 4); this assumption can be relaxed to further requirements that A can partially verify [3].

Refining Lemma 1 by this result yields **Theorem 1**: if the messages m and m' appear in the traffic and K is confidential, then B *knows* K. Therefore, If A ever gets hold of m and m' and verifies the confidentiality of K, then she can apply the theorem and deduce her peer's knowledge of K. I have proved a symmetric guarantee for B.

Similarly, in Model 2, if the events

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

occur on a trace *evs*, and K is confidential, I can prove that

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

occurs on *evs*, and then that event (2) also occurs on *evs*. Therefore, Lemma 2 can be refined by this result producing **Theorem 2**, which is a guarantee for A: if A receives the messages m and m', and verifies K to be confidential, then B *knows* K.

5 Discussion

Theorems 1 and 2 are practically equivalent because they become applicable to A at the same instance of time, i.e. when she gets hold of the messages m and m' (clearly, A must first decrypt m in order to extract the session key K to decrypt m'). Both theorems establish the same result, which may be interpreted as *B's non-repudiation of knowledge of K with A* because it forbids B to mislead A about B not knowing K.

The only difference is that, since Model 1 does not contain message reception, Theorem 1 must refer to some earlier time: the instance when the messages appeared in the traffic. Consequently, it could be argued that Theorem 1 is somewhat stronger because it enforces its result earlier than Theorem 2 does.

The requirement of confidentiality of the session key is necessary to prove that a certain event involving such key could only occur if some other event did previously occur on the same trace. As described above, this strategy was followed for proving both theorems, which therefore must assume a confidential session key.

Consider a protocol whose last message delivers a session key, say, to B. In this scenario, Model 1 cannot help because no Issues property can be proved. By Model 2, I could still enforce a result of the form of Lemma 2 but not of Theorem 2 because there exists no event that implies B's reception of the last message of the protocol. Therefore, the result of Lemma 2 cannot be made available to A.

Now, think of a protocol which delivers a session key to B and then terminates with B sending a message to A. Even if the message does not contain the session key, I can exploit this last event to establish a result of the form of Theorem 2. Therefore, Model 2 seems to have a broader scope.

The two models could be easily combined to capture a broader notion of knowledge, but I believe that Model 2 can elegantly cope with most realistic circumstances. Both models may be used to reason about knowledge of any message items.

Acknowledgements. Special thanks to Larry Paulson for supervising my Ph.D.

References

1. M. Abadi, M. Burrows, C. Kaufman, B. Lampson. Authentication and Delegation with Smart-cards. DIGITAL Technical Report 67, California, 1990.
2. G. Bella. Enhancing the Inductive Approach by Message Reception. *Technical Report No. 460*, Cambridge University, Computer Laboratory, 1999.
3. G. Bella. Are Timestamps Worth the Effort? A Formal Treatment. *Technical Report No. 427*, Cambridge University, Computer Laboratory, 1998.
4. 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.
5. G. Lowe. A Hierarchy of Authentication Specifications. Proc. of *Tenth IEEE Computer Security Foundations Workshop*, 1997.
6. L. C. Paulson. The Inductive Approach to Verifying Cryptographic Protocols. *Journal of Computer Security*, 6:85-128, 1998.