Holistic Analysis of Mix Protocols

Giampaolo Bella
DMI, Università di Catania, Italy
STRL, De Montfort University, UK
giamp@dmi.unict.it

Denis Butin
School of Computing
Dublin City University, Ireland
denis.butin@computing.dcu.ie

David Gray
School of Computing
Dublin City University, Ireland
david.gray@computing.dcu.ie

Abstract

Security protocols are often analysed in isolation as academic challenges. However, the real world can require various combinations of them, such as a certified email protocol executed over a resilient channel, or the key registration protocol to precede the purchase protocols of Secure Electronic Transactions (SET).

We develop what appears to be the first scalable approach to specifying and analysing mix protocols. It expands on the Inductive Method by exploiting the simplicity with which inductive definitions can refer to each other. This lets the human analyst study each protocol separately first, and then derive holistic properties about the mix.

The approach, which is demonstrated on the sequential composition of a certification protocol with an authentication one, is not limited by the features of the protocols, which can, for example, share message components such as cryptographic keys and nonces. It bears potential for the analysis of complex protocols constructed by general composition of others.

Index Terms

Inductive Method; Theorem Proving; Isabelle; Security;

1. Introduction

The formal analysis of security protocols is currently so mature that its conceptual challenges appear to have lost much of their appeal with the international research community. This augments the quest towards making the analysis fully automatic, arguably a very worthy goal.

1.1. Motivations

The present manuscript demonstrates that the challenges of protocol analysis are still significant at least upon those protocols that are run *with* other protocols. In particular, security protocols are typically sequenced, as is the case of protocols for public-key registration, certification and actual use. They are often executed on top of one another in a stack fashion, as is the case of an SSL session taking place over an IPSec channel. Protocols may also be interleaved, perhaps with malicious aims, as is the case of a purchase transaction that is entwined with a banking session. We advocate that protocol analysis become more precise (hence more reliable) upon these protocols, and hence make explicit a number of preconditions whose validity is given for granted.

We term *mix protocol* each combination of protocols by means of sequencing, stacking or interleaving. The analysis of mix protocols still appears out of reach for the automatic protocol analyser ProVerif [1] but not for Scyther [2], which can handle some. However protocols are composed, their interactions make isolated security analysis hazardous. Intuitively, protocols sharing components may influence security guarantees in unforeseen ways. In practice, the risk of maliciously interleaving the sessions of two different protocols was demonstrated by Cremers, the author of Scyther [3], who exposed a number of multi-protocol attacks. His tool is however limited to "specific classes of protocols that can be composed in certain ways" [4], namely classes whose protocols are "strongly independent". In fact, "ciphertexts, signatures, and message authentication tags originating in one protocol set will never be accepted by the other protocol set and vice versa". By contrast, our approach can tackle protocols that interact a lot and in fact depend on each other, such as a public-key authentication protocol depending on a certification one by using its certificates — notably, the two protocols can be specified and studied separately, but they can use the same elements, such as keys and certificates.

Also, the technicalities of ad hoc protocol tools may somewhat hide the operational features of mix protocols and limit their understanding to non-practitioners of the software. Hence the need to reason about these protocols using a mathematically-rooted and well-understood language such as induction, which is not constrained by completeness issues.

1.2. Findings

The main finding of this paper is how to formally analyse mix protocols using the Inductive Method [5], [6], an estab-

Research supported by the SFI grant 08/RFP/CMS1347.

lished approach to protocol analysis based on mathematical induction and the support of the interactive theorem prover Isabelle. Although Isabelle requires user specialisation to conduct the proofs, the findings are intuitive thanks to the simplicity of induction. Therefore, they can be easily understood, for example, by mathematicians approaching protocol analysis without interest in learning specialised tools, also because most proofs can be reproduced by pen and paper, depending on their size.

In particular, it is found that mix protocols can be specified holistically by using more than one inductive protocol definition, precisely one per protocol. In case of sequencing, a protocol step can be premised with particular conditions about a previous protocol. Typically, such conditions express the achievement of the main goals of the previous protocol, such as the distribution of a session key or the authentication of a peer. In abstract, logical terms, the case of stacking is the same, as the necessary premise conditions refer to the underlying protocol. In case of interleaving, the two (n in general) protocols refer to each other using the same mechanism, a feature that enforces the advancement of a protocol only upon condition that the other protocol advances too. This approach bears the potential to scale up easily to the specification of protocols obtained by composition.

This holistic approach is demonstrated in this paper upon a mix protocol built by sequencing. First, a general certification protocol whereby an agent receives from a certification authority a certificate for herself and her peer is defined and studied inductively. Then, an authentication protocol is tackled. Notably, it is based on its peers' knowledge of the necessary certificates, hence on the successful completion of the underlying certification protocol. For the sake of demonstration, we chose the best-known authentication protocol, due to Needham and Schroeder, which has rarely been analysed in its full version including certification. While Meadows analysed it as a monolithic entity [7], we will treat certification separately from the remaining protocol but derive holistic guarantees about their sequential combination.

The second finding is a general treatment of certification, which can be reused for all public-key protocols tackled so far. Agents that had to refer to each public key as if their owner was magically known can now use a public key accordingly to the corresponding certificate, which is signed by the authority. By reflecting exactly what happens in the real world, this reaches our aim of a more precise and reliable analysis of the authentication protocol. The depth of the certificate chain is kept to one level for simplicity, but it can be naturally generalised to many levels.

1.3. Paper summary

This paper continues by presenting the certification and authentication protocols featured in a running example (§2). After a short reminder on the formal tools that are adopted, Isabelle/HOL and the Inductive Method (§3), the main guarantees for our case study are discussed (§4). More details on the formal specifications and proofs follow (§5). Some conclusion terminate the treatment, and outline the potential for future applications arising from the scalability of the approach (§6).

2. Running example

Our research begins with the analysis of the generic certification protocol in Figure 1. The standard protocol notation is adopted, and the reader's familiarity with it is assumed. An agent A contacts a certification authority *CA* to obtain public-key certificates for her and her intended peer B. In a subsequent protocol, A may use her certificate to forward it to her peer, and needs her peer's to meet a very basic requirement: knowing what public key to use with her peer to ensure that only he can decrypt her traffic. Public knowledge of the *CA*'s public key is assumed, but could be treated as the outcome of a previous protocol in the sequence.

$$\begin{array}{lrcl} 1. & A \longrightarrow CA & : & A, B \\ 2. & CA \longrightarrow A & : & \{\!|K_a, A|\!\}_{K_{CA}^{-1}}, \{\!|K_b, B|\!\}_{K_{CA}^{-1}} \end{array}$$

Figure 1. A generic certification protocol

The most published protocol ever is (yet again) quoted in Figure 2 for the reader's convenience. It is the public-key Needham-Schroeder protocol [8] with Lowe's fix [9] of repeating B's identity next to the nonce pair in a message. This is the full version that includes the certification steps, which are, irrespectively of their importance, usually simplified away. We present the protocol with appropriate line spacing in order to emphasise that the first two steps provide the initiator with her peer's certificate, as the trusted server S is offering the certification service. Then, steps 4 and 5 are the homologous for B.

$$\begin{array}{lrcl} 1. & A \longrightarrow S & : & A, B \\ 2. & S \longrightarrow A & : & \{\!|K_b, B|\!\}_{K_S^{-1}} \\ 3. & A \longrightarrow B & : & \{\!|N_a, A|\!\}_{K_b} \\ 4. & B \longrightarrow S & : & B, A \\ 5. & S \longrightarrow B & : & \{\!|K_a, A|\!\}_{K_S^{-1}} \\ 6. & B \longrightarrow A & : & \{\!|N_a, N_b, B|\!\}_{K_a} \\ 7. & A \longrightarrow B & : & \{\!|N_b|\!\}_{K_b} \end{array}$$

Figure 2. The full public-key Needham-Schroeder protocol with Lowe's fix

The four steps described above signify the peers' respective execution of a specific certification protocol, following

a registration protocol whereby the public key is securely stored with the authority. The full protocol is thus obtained by sequencing a certification protocol with an authentication protocol consisting of the remaining three steps, 3, 6 and 7, those that are commonly though simplistically addressed as Needham-Schroeder protocol. Without loss of generality, that certification protocol could be our generic one (Figure 1). In consequence, a reliable formal analysis of the full protocol demands a formal approach that can deal with mix protocols constructed by sequencing. This is what the sequel of this manuscript shall demonstrate.

3. Method

The method that our work adopts is the Inductive Method. This Section only provides a glimpse at it to serve as a reminder to those who already are familiar with it.

Isabelle [10] is the latest descendant of a long line of theorem provers. It is a generic theorem prover, based on ML, with a wide field of applications. Many logics are supported, since Isabelle is general-purpose; the most widely used one is Isabelle/HOL, allowing predicates to be formalized and proved in higher-order logic. Proofs are interactive: even though simplification and classical reasoning tools are available, the user must still guide the chain of proofs and sometimes prove subgoals manually. A file-based theory hierarchy provides a flexible starting point for proofs. Every theory file inherits all theorems and proofs from parent theories.

Isabelle/HOL's application to security protocol verification, called the Inductive Method, was introduced by Paulson [11] and refined by Bella [6]. Mathematical induction is used to model protocol steps, define security goals and prove them. It was applied to a variety of protocols, even large ones such as the SET suite [12]. Protocol steps, a Dolev-Yao type threat model and security guarantees are modelled inductively. Security protocols formalised in Isabelle typically import the theory file *Public*, specifying cryptographic keys and initial states. Its parent file is *Event*, which mainly describes network event datatypes. The third theory specific to protocol analysis is *Message* ; it describes the structure of messages and agents and is imported by *Event*.

In the Inductive Method, the number of agents and concurrent protocol sessions are unbounded, which allows replay and interleaving attacks to be detected. Normal agents are honest, and merely follow the protocol. Their number is not bounded. A trusted server and a dishonest agent, the Spy, are also present.

The Spy is a peer that can act legally, like a friendly agent, but that has additional powers too. Like in the Dolev-Yao model, which she embodies, the Spy in the Inductive Method can forge and send new messages combining any parts of messages previously sent over the network. She can transmit existing ciphertext, but cannot create new ciphertext without possessing the corresponding encryption key. This set of abilities is modelled by the following inductive rule:

$$\mid \mathit{Fake}: [\![\mathit{evsf} \in \mathit{ns\_public};\ X \in \mathit{synth}\ (\mathit{analz}\ (\mathit{knows}\ \mathit{Spy}\ \mathit{evsf}))]\!] \Longrightarrow \mathit{Says}\ \mathit{Spy}\ B\ X\ \#\ \mathit{evsf} \in \mathit{ns\_public}$$

Security properties are established by inductive reasoning over lists of network events called traces. Every protocol step is modelled as an inductive rule with pre- and postconditions. The protocol model is the set of all admissible traces under those rules. A security property is considered proven if it holds for all those possible traces. The agents' knowledge is derived from the traces.

Network events resulting from the protocol are represented by *Says*, *Gets* and *Notes*. *Says A B X* means that agent *A* sends the message *X* to agent *B*. Since delivery is not guaranteed, this does not automatically imply *B* receiving *X*. *Gets B X* models this very fact. The event *Notes A X* models internal storage of the message *X* by agent *A*.

Asymmetric cryptographic key-pairs are declared as *priSK* and *pubSK* for signature and *priEK* and *pubEK* for encryption. When the distinction is not used, one can simply write *priK* and *pubK*. The function *invKey* maps an asymmetric key to its inverse. Before a protocol begins, agents often already know some keys or other elements; this is formalized by the *initState* function, which maps an agent to a set of messages.

The operator *analz* formalises breaking up messages into their elements. This includes decoding ciphertexts when the key is available, but not cryptanalysis. Cryptography is seen as a black-box, and considered perfect as long as the key doesnt leak. On the other hand, the *parts* operator applied to a set of messages returns all building blocks from those messages, even ciphertext for which the key is not known to the agent. *synth* formalises the creation of new messages from available components. Encryption, which is formalised by the function *Crypt*, is considered unbreakable unless the adequate key is leaked.

The function *knows Spy* maps a list of network events to a set of messages. Those message are the knowledge that the Spy acquired from that particular set of events. Since the Spy intercepts all exchanged messages, this set contains the network traffic for the given trace. A generalization of *knows Spy* is the function *knows*, taking as parameters an agent and a trace and outputting a set of messages: the agents knowledge. Agents know what they send, note, or receive.

Compromised agents reveal their private keys and what they *Note* to the Spy, and are denoted by the set *bad*. The Spy is herself part of *bad*.

The function *used* maps a list of events to a set of messages, and allows us to reason about freshness. A message component is defined as fresh if it is present in no initial state and if it has never been part of a *Says* or *Notes* event.

When proving a security property result about a protocol, we first give it a name, preceded by the command *theorem*

or *lemma*. The result proper is then stated between double quotes, followed by the proof. Arguments such as *auto*, *simp_all* or *blast* make some proof steps automatic, but user guidance is needed for intricate parts.

4. Summary of the findings

The generic certification protocol (Figure 1) has been analysed using the standard techniques available in the Inductive Method. Despite the presence of an active attacker, the standard Dolev-Yao [13], the protocol succeeds in establishing its intended security properties. The corresponding guarantees are summarised and discussed here, while their proofs are deferred (§5).

Theorem 1 (Says_CA_cert)**.** *The certification protocol establishes the following security properties.*

- *A message that the certification authority sends contains two different well-formed certificates.*
- *Each certificate contains the key that is addressed as public key of the agent that the certificate mentions.*

This Theorem holds because the authority cannot misbehave. Its structure is not new because other protocols analysed so far rest on similar assumptions. However, its significance is innovative. It formalises for the first time that a bitstring contained in a certificate can be addressed as public key of the agent mentioned beside it. Formally, a generic key K found inside a certificate next to a generic agent name A implies that $K = pubEK\ A$. To enable the protocol participants to appeal to this theorem, another guarantee is needed.

Theorem 2 (cert_authentic_agent)**.** *The certification protocol establishes the following security properties.*

- *A well-formed certificate that a generic agent obtains originated with the certification authority.*
- *The authority may have sent it as either first or second component of its message.*

By combining this Theorem with the previous, a generic agent can conclude that the mentioned key is the public key of the agent mentioned next to it. Despite the brevity of the statement, theorems of this form have never been proved before. It generalises to a generic agent the existing authenticity guarantees [6] confirming the originator of messages available to the attacker. This generalisation has required novel proof strategies involving a number of case splits, which are detailed later.

The main findings about the authentication protocol stem from each agent's check of their peer's certificate prior to sending the relevant protocol message, as we shall see (§5). The main confidentiality guarantees about the exchanged nonces can thus be expanded by stating the form of the encrypting key via an appeal to Theorem 1.

Theorem 3 (Spy_not_see_NA)**.** *The authentication protocol, if executed after the certification protocol, establishes the following security properties.*

- *The first message that an initiator agent sends to a responder agent while both conform to the protocol is such that:*
  - *it contains a confidential initiator's nonce;*
  - *it is encrypted by the responder's public key.*

Theorem 4 (Spy_not_see_NB)**.** *The authentication protocol, if executed after the certification protocol, establishes the following security properties.*

- *The reply that a responder agent sends to an initiator agent while both conform to the protocol is such that:*
  - *it contains a confidential responder's nonce;*
  - *it is encrypted by the initiator's public key.*

These guarantees confirm what happens in the real world. When honest agents engage in the authentication protocol, each of them does not have to blindly assume that they are using the right key. By contrast, each inherits a guarantee from the preceding certification protocol that they precisely are using the public key belonging to the intended peer. This level of detail was not available before our work.

5. Details of the findings

5.1. Specifications

The certification protocol can be studied using traditional techniques. The inductive specification of its main event is quoted here.

$$
\begin{aligned}
\mid\ & Cert2\colon [\![evsc2 \in cert;\ Gets\ CA\ \{\!|Agent\ A, Agent\ B|\!\} \in set\ evsc2;\\
& \quad A \neq B]\!]\\
& \quad \Longrightarrow Says\ CA\ A\\
& \qquad \{\!|Crypt\ (priSK\ CA)\ \{\!|Key\ (pubEK\ A), Agent\ A|\!\},\\
& \qquad\ Crypt\ (priSK\ CA)\ \{\!|Key\ (pubEK\ B), Agent\ B|\!\}|\!\}\\
& \qquad\ \#\ evsc2 \in cert
\end{aligned}
$$

It can be seen that upon receiving a valid certificate request, *CA* replies to the agent first quoted in the message by sending two certificates: one for each agent quoted. The authority only checks that it is issuing certificates for two different agents. A signature by *CA* is indicated by *Crypt (priSK CA)*.

The specification phase can now tackle the authentication protocol, and can refer to the certification one. Let *NA* be a fresh nonce and assume a certificate mentioning agent B is part of A's knowledge derived from the certification protocol. Assume also that A obtained B's certificate on some trace of the certification protocol. Then the first message of the authentication protocol is sent by A to B: the concatenation of *NA* and of the sender's name, all encrypted with the key found in the certificate for B.

$$
\begin{aligned}
&| \mathit{NS1}: \ [\![\mathit{evs1} \in \mathit{ns\_public}; \mathit{Nonce\ NA} \notin \mathit{used\ evs1}; \mathit{evsca} \in \mathit{cert};\\
&\quad \mathit{Crypt}\ (\mathit{priSK\ CA})\ \{\!|\mathit{Key\ K}, \mathit{Agent\ B}|\!\}\\
&\quad \in \mathit{parts}(\mathit{knows\ A\ evsca})]\!]\\
&\quad \Longrightarrow \mathit{Says\ A\ B}\ (\mathit{Crypt\ K}\ \{\!|\mathit{Nonce\ NA}, \mathit{Agent\ A}|\!\})\\
&\qquad \#\ \mathit{evs1} \in \mathit{ns\_public}
\end{aligned}
$$

Notably, it is the first time that the specification of a protocol with the Inductive Method needs to make assumptions on traces of two different protocols, here *evsca* from the protocol specified by *cert*, and *evs1* from the protocol being specified by *ns_public*. Precisely, the assumption on *evsca* serves to bind the key that *A* uses to build the new message.

Similarly, if *B* has received a message of format *NS1* and knows a certificate for *A*, then he picks another fresh nonce *NB* and sends it, encrypted, to *A*, along with the previously received nonce and *B*'s identity.

$$
\begin{aligned}
&| \mathit{NS2}: \ [\![\mathit{evs2} \in \mathit{ns\_public}; \mathit{Nonce\ NB} \notin \mathit{used\ evs2}; \mathit{evscb} \in \mathit{cert};\\
&\quad \mathit{Gets\ B}\ (\mathit{Crypt}\ (\mathit{pubEK\ B})\ \{\!|\mathit{Nonce\ NA}, \mathit{Agent\ A}|\!\})\\
&\quad \in \mathit{set\ evs2};\\
&\quad \mathit{Crypt}\ (\mathit{priSK\ CA})\ \{\!|\mathit{Key\ K}, \mathit{Agent\ A}|\!\}\\
&\quad \in \mathit{parts}(\mathit{knows\ B\ evscb})]\!]\\
&\quad \Longrightarrow \mathit{Says\ B\ A}\ (\mathit{Crypt\ K}\ \{\!|\mathit{Nonce\ NA}, \mathit{Nonce\ NB},\\
&\qquad\qquad \mathit{Agent\ B}|\!\})\ \#\ \mathit{evs2} \in \mathit{ns\_public}
\end{aligned}
$$

The explicit reference to traces belonging to the two protocols is visible also in this case. In line with the previous step, the assumption on *evscb* serves to bind the key that *B* uses to build the new message.

Finally, if *A* has sent the first message and received the the second message, he sends a new message to *B*, quoting the nonce *B* provided, and using the same encryption key as in the first message.

$$
\begin{aligned}
&| \mathit{NS3}: \ [\![\mathit{evs3} \in \mathit{ns\_public};\\
&\quad \mathit{Says\ A\ \ B}\ (\mathit{Crypt\ K}\ \{\!|\mathit{Nonce\ NA}, \mathit{Agent\ A}|\!\}) \in \mathit{set\ evs3};\\
&\quad \mathit{Gets\ A}\ (\mathit{Crypt}\ (\mathit{pubEK\ A})\ \{\!|\mathit{Nonce\ NA},\\
&\qquad\qquad \mathit{Nonce\ NB}, \mathit{Agent\ B}|\!\}) \in \mathit{set\ evs3}]\!]\\
&\quad \Longrightarrow \mathit{Says\ A\ B}\ (\mathit{Crypt\ K}\ (\mathit{Nonce\ NB}))\ \#\ \mathit{evs3} \in \mathit{ns\_public}
\end{aligned}
$$

According to the principle of *guarantee availability* [14], only one of the encrypting keys can be specified. Because this rule defines an action of *A*'s, she can check that the message she gets is sealed under her public key. By contrast, she cannot check that the message she sent her peer was encrypted with his public key: this must be proved in the model. In fact, such result holds because the event is traced back to when it was issued in the first protocol step, when access to the right certificate was assumed.

5.2. Proofs

This Section outlines the actual theorems proved in Isabelle to support less formal version discussed above (§4).

Theorem 1 derives from the combination of the two following theorems.

theorem *Says_CA_cert1*:
$$
\begin{aligned}
&[\![\mathit{Says\ CA\ A}\ \{\!|\mathit{Crypt}\ (\mathit{priSK\ CA})\ \{\!|\mathit{Key\ K}, \mathit{Agent\ A}|\!\},\\
&\qquad \mathit{certB}|\!\} \in \mathit{set\ evs}; \mathit{evs} \in \mathit{cert}]\!]\\
&\Longrightarrow K = \mathit{pubEK\ A}
\end{aligned}
$$

The statement can be read as follows. For any sequence of events following the certification protocol, if *CA* sends a message containing a specific certificate and another component, then the mentioned key is the public encryption key of the agent mentioned beside the key. A version for the other certificate can be proved too.

theorem *Says_CA_cert2*:
$$
\begin{aligned}
&[\![\mathit{Says\ CA\ A}\ \{\!|\mathit{certA}, \mathit{Crypt}\ (\mathit{priSK\ CA})\ \{\!|\mathit{Key\ K}, \mathit{Agent\ B}|\!\}|\!\}\\
&\ \in \mathit{set\ evs}; \mathit{evs} \in \mathit{cert}]\!]\\
&\ \Longrightarrow K = \mathit{pubEK\ B} \wedge A \neq B
\end{aligned}
$$

It can be observed that this Theorem, contrarily to the previous, specifies the agent pair, hence it can conclude that they are different by leveraging on the assumption stated by rule *Cert2* seen above (§5.1).

Theorem 2 rests on the following innovative statement.

theorem *cert_authentic_agent*:
$$
\begin{aligned}
&[\![\mathit{Crypt}\ (\mathit{priSK\ CA})\ \{\!|\mathit{Key\ K}, \mathit{Agent\ B}|\!\} \in \mathit{parts}(\mathit{knows\ A\ evs});\\
&\ \mathit{evs} \in \mathit{cert}]\!]\\
&\ \Longrightarrow (\exists\ D\ \mathit{certB}.\\
&\qquad \mathit{Says\ CA\ D}\ \{\!|\mathit{certB}, \mathit{Crypt}\ (\mathit{priSK\ CA})\ \{\!|\mathit{Key\ K}, \mathit{Agent\ B}|\!\}|\!\}\\
&\qquad \in \mathit{set\ evs})\\
&\qquad \vee\\
&\qquad (\exists\ \mathit{certB}.\\
&\qquad \mathit{Says\ CA\ B}\ \{\!|\mathit{Crypt}\ (\mathit{priSK\ CA})\ \{\!|\mathit{Key\ K}, \mathit{Agent\ B}|\!\}, \mathit{certB}|\!\}\\
&\qquad \in \mathit{set\ evs})
\end{aligned}
$$

This result requires particular attention because it is the first significant fact proved *upon assuming* something about the knowledge of a generic agent. By contrast, existing proofs only consider the knowledge of the Spy. Here, *A* can be either the Spy or a regular agent. As a consequence, the proof features two successive inductions on the certification protocol model. First, assume *A* is the Spy. In this case, induction and simplification leave us with two remaining subgoals: the *Fake* case, which embeds the threat model, and the subgoal arising from the second protocol step. Those two cases are treated by classical methods, *spy_analz* and *blast*.

If *A* is an honest agent, that is different from the Spy, the proof is more intricate and was unexplored before the present effort. We must perform a number of case splits and reason about how protocol events modify agent knowledge. In existing Inductive Method theories, simplification lemmas are provided to deal with changes to the Spy's knowledge upon occurrence of protocol events, but the case of regular agents is not spelled out fully. The reasoning is therefore performed along the proof by expanding the definition of *knows* as appropriate.

The case of the *Reception* case is particularly interesting. We first perform a case split on whether a certificate for *B* appeared in the traffic — that is, it exists in the Spy's knowledge. If such is the case, a traditional authenticity result about the certificate (*cert_authentic*, here omitted for brevity) can be applied and allows us to conclude. Otherwise

our subgoal still features the premise *Crypt (priSK CA) {Key K, Agent B} ∈ parts (knows A (Gets Ba X # evsr))* and we must differentiate between the scenarios *A≠Ba* and *A=Ba*. In the former case, the *Gets* event cannot influence *A*'s knowledge because honest agent do not see all traffic, hence we obtain a contradiction. Else, *Crypt (priSK CA) Key K, Agent B* must be in *parts (insert X (knows A evsr))*. Since it is not in *parts (knows A evsr)*, it must be in *parts{X}*, but *X* appears in a *Says* event — hence it must be known to the Spy, a contradiction.

Theorem 3 can now be explained.

theorem *Spy_not_see_NA*:
⟦*Says A B* (*Crypt K* {|*Nonce NA*, *Agent A*|}) ∈ *set evs*;
A ∉ *bad*; *B* ∉ *bad*; *evs* ∈ *ns_public*⟧
⟹ *Nonce NA* ∉ *analz* (*knows Spy evs*) ∧ *K* = *pubEK B*

This establishes the confidentiality of the initiator's nonce in the authentication protocol. It resembles the guarantee that can be found in the Isabelle repository [15]: if the protocol step *NS1* takes place and *A* and *B* are honest, then the nonce from *NS1* remains secret. In addition, it specifies *K* to be *B*'s public key. Notably, this requires appeals to the guarantees about the other protocol, the certification one. Thanks to the premise about trace *evsca* in rule *NS1* seen above (§5.1), theorem *cert_authentic_agent* can be applied. Then, the combination of *Says_CA_cert1* and *Says_CA_cert2* pinpoints the contents of the certificates.

Similarly, Theorem 4 derives from the following result.

theorem *Spy_not_see_NB* :
⟦*Says B A* (*Crypt K* {|*Nonce NA*, *Nonce NB*, *Agent B*|}) ∈ *set evs*;
A ∉ *bad*; *B* ∉ *bad*; *evs* ∈ *ns_public*⟧
⟹ *Nonce NB* ∉ *analz* (*knows Spy evs*) ∧ *K* = *pubEK A*

This result can be commented similarly to the previous. However, the results about the certification protocol, starting with *cert_authentic_agent*, can be applied thanks to the premise about trace *evscb* in rule *NS2* seen above (§5.1).

6. Conclusions

We have described the formal modelling of mix protocols and their holistic analysis in the interactive theorem prover Isabelle/HOL. Sequenced, stacked and interleaved protocols can be specified and verified in a framework with rigorous foundations using inductive mathematical reasoning. In our running example, we have analysed a sequence featuring a generic key certification protocol and a simple authentication protocol. For the specific case of key certification, our approach is scalable to a full PKI model featuring multiple levels of trust. The adopted strategy for mix protocol specification translates into a proving process featuring novel situations. Those could be tackled effectively with the mechanical support of the interactive theorem prover. While automatic protocol analysers are making progress, more general tools like Isabelle/HOL and the Inductive Method provide invaluable flexibility for reasoning in detail about common and uncommon protocol combinations. Intricate protocol interactions and mixes of more than two protocols are our natural next objects of study.

References

[1] B. Blanchet, "An efficient cryptographic protocol verifier based on Prolog rules," 1998, pp. 82–96.

[2] C. Cremers, "The Scyther Tool: Verification, falsification, and analysis of security protocols," in *Proc. of the 20th International Conference on Computer Aided Verification (CAV 2008)*, ser. LNCS 5123. Springer, 2008, pp. 414–418.

[3] ——, "Feasibility of multi-protocol attacks," in *Proc. of The First International Conference on Availability, Reliability and Security (ARES)*. Vienna, Austria: IEEE Computer Society, 2006, pp. 287–294.

[4] S. Andova, C. Cremers, K. Gjøsteen, S. Mauw, S. Mjølsnes, and S. Radomirović, "A framework for compositional verification of security protocols," *Information and Computation*, vol. 206, pp. 425–459, February 2008.

[5] L. C. Paulson, "The inductive approach to verifying cryptographic protocols," vol. 6, pp. 85–128, 1998.

[6] G. Bella, *Formal Correctness of Security Protocols*, ser. Information Security and Cryptography. Springer, 2007.

[7] C. Meadows, "Analyzing the needham-schroeder public-key protocol: A comparison of two approaches," in *ESORICS*, 1996, pp. 351–364.

[8] R. M. Needham and M. D. Schroeder, "Using encryption for authentication in large networks of computers," vol. 21, no. 12, pp. 993–999, 1978.

[9] G. Lowe, "Breaking and fixing the Needham-Schroeder public-key protocol using CSP and FDR," ser. LNCS 1055, T. Margaria and B. Steffen, Eds., 1996, pp. 147–166.

[10] L. C. Paulson, *Isabelle: A Generic Theorem Prover*, ser. LNCS 828, 1994.

[11] ——, "Proving properties of security protocols by induction," 1997, pp. 70–83.

[12] G. Bella, F. Massacci, L. C. Paulson, and P. Tramontano, "Formal verification of cardholder registration in set," ser. LNCS 1895, F. Cuppens, Y. Deswarte, D. Gollmann, and M. Waidner, Eds., 2000, pp. 159–174.

[13] D. Dolev and A. Yao, "On the security of public-key protocols," *IEEE Transactions on Information Theory*, vol. 2, no. 29, pp. 198–208, 1983.

[14] G. Bella, "The principle of guarantee availability for security protocol analysis," *Int. J. Inf. Secur.*, vol. 9, pp. 83–97, April 2010.

[15] G. Bella, F. Blanqui, and L. C. Paulson, *On-line Repository of Protocol Proofs*, As from Isabelle 2006, `http://isabelle.in.tum.de/library/HOL/Auth`.