Int. J. Inf. Secur. (2010) 9:83–97
DOI 10.1007/s10207-009-0097-y

REGULAR CONTRIBUTION

The principle of guarantee availability for security protocol analysis

Giampaolo Bella

Published online: 9 December 2009

Abstract Conformity to prudent design principles is an established approach to protocol correctness although it is not free of limitations. We term *goal availability* a design principle that is often implicitly followed, prescribing protocols to aim at principal-centric goals. Adherence to a design principle is normally established through protocol analysis that is an evaluation of whether a protocol achieves its goals. However, the literature shows that there exists no clear guidance on how to conduct and interpret such an analysis, a process that is only left to the analyzer's skill and experience. Goal availability has the desirable feature that its supporting protocol analysis can be precisely guided by what becomes a principle of realistic analysis, which we call *guarantee availability*. It prescribes that the outcome of the analysis, which is the set of guarantees confirming the protocol goals, be practically applicable by the protocol participants. In consequence, the guarantees must be based on assumptions that the principals have the capacity to verify. Our focus then turns entirely to protocol analysis, because an analysis conforming to guarantee availability signifies that the analyzed protocol conforms to goal availability. Existing analysis of (both classical and deployed) protocols has been reconsidered with the aim of studying their conformity to guarantee availability. Some experiments clarify the relationships between goal availability and the existing design principles, with particular reference to explicitness. Other experiments demonstrate that boosting an analysis with guarantee availability generally makes it deeper, unveiling additional protocol niceties that depending on the analyzer's skills may remain overseen otherwise. In particular, an established claim about a protocol (made using a well-known formal method) can be subverted.

Keywords Formal analysis · Theorem proving · Network protocol · Trust

1 Introduction

Security protocols are fundamental security measures for distributed communications. They are designed to achieve a set of goals, such as authentication and confidentiality, but experience shows that they may unexpectedly fail. Protocol analysis, which is the process of establishing whether a protocol meets its goals, has therefore become particularly important. A popular approach to protocol analysis is to adopt some formal method to develop abstract protocol models and variously study them. Among the seminal methods of formal analysis is the BAN logic and its derivations, which formulate as logical predicates the beliefs that the protocol principals derive while they are executing the protocol, and treat these beliefs using dedicated inference rules [12,13]. Another method entirely relies on mathematical induction both for specification and for verification purposes [7,27]. Yet, another one uses the process algebra CSP to get compact protocol models, and adopts various proof techniques to study a variety of protocol goals [28,29]. This list is truncated by necessity.

Another popular approach to protocol analysis advances a number of general principles for prudent protocol design and informally studies conformity of protocols to them. Although these principles are neither necessary nor sufficient for a protocol to meet its goals, they have contributed to unveiling a

G. Bella (✉)
Dip. Matematica e Informatica, Università di Catania,
Viale A.Doria 6, 95125 Catania, Italy
e-mail: giamp@dmi.unict.it

G. Bella
Software Technology Research Lab, De Montfort University,
The Gateway, Leicester LE1 9BH, UK

number of protocol subtleties [2,4]. Some design principles concern, for example, the appropriate use of encryption, and point out that "extra encryption" is not necessarily the same as "extra security". Others pertain to timeliness, pointing out how to use a nonce for freshness purposes. But the principles that have been most often appealed to perhaps are those about *explicitness*. They prescribe that each message be explicit about its contents, otherwise the principals would be forced to heuristic interpretations that typically become points of convergence of attackers' efforts. It is recommended to take a cautionary look at these principles [32].

Our research proceeds from the delineation of a principle of protocol design, *goal availability*, requiring a protocol to specifically aim at principal-centric goals rather than protocol-centric (also said god-centric) ones. For example, key distribution that a protocol delivers a session key to its peers has a well-known principal-centric version called *key confirmation of Y to X*, holding "if entity X is assured that the second entity Y actually has possession of a particular secret key" [31], which in this paper is always a session key. The relationship of goal availability with the existing design principles, explicitness in particular, is not obvious, and hence this paper unfolds it through examples. It boils down to the topic of the principle, which with goal availability is the protocol goals rather than the features of the protocol messages.

Following the conventional practice of checking adherence of a protocol to a principle by informal analysis, we soon realize that with goal availability, the supporting analysis can be given specific guidance, which we define as a principle of realistic protocol analysis. Named *guarantee availability*, its abstract version simply prescribes the analysis to focus on principal-centric goals. More specifically, guarantee availability prescribes the development of guarantees that protocol participants can use in practice, which are guarantees based on assumptions that the protocol participants can verify by themselves to a large extent in some adequate threat model. Our main finding is that guarantee availability turns out to significantly deepen the analyses. For example, we shall see how this principle helps to interpret protocol guarantees systematically and therefore answer questions of the form: is this guarantee sufficient to let us conclude that the protocol meets that specific goal?

Using formal methods is widely accepted as the gold standard approach to protocol analysis. Our research originally combines the prudent principle approach with the formal method approach to protocol analysis: it provides the evidence in support of the new principle of design, goal availability, by means of formal, rather than informal analysis. Guarantee availability is profitably applied to such formal analysis.

The findings of an analysis that fail to conform to guarantee availability may generally be of little use to the principals because the principals may be unable to appeal to them. For example, suppose that an electronic merchant vows in front of a notary public that he will ship the goods if he receives the relevant payment. If an electronic client pays for the goods but fails to receive them, she cannot easily sue the merchant on the sheer basis of his promise. A judge may be seen as the analyzer of the controversy. He would face the problem of verifying whether the precondition of the promise holds, namely that the merchant receives the payment, for the promise to be beneficial to the controversy. The formal analysis of protocols often produces similar conditional guarantees. Guarantee availability equally insists that a formal guarantee is beneficial to a principal only if the principal can verify (most of) its preconditions. Therefore, although our conceptual contribution is a pair of principles, one for protocol design and one for analysis, in practice we can only work with the latter: an analysis conforming to guarantee availability indicates that the analyzed protocol conforms to goal availability.

Moreover, our experiments show that adherence to guarantee availability adds significance to an analysis by revealing additional protocol niceties. In particular, we argue that checking conformity of the analysis of the asymmetric cryptography Needham–Schroeder protocol to guarantee availability would have anticipated the discovery of the lack of explicitness leading to Lowe's attack. A similar finding arises from the Shoup–Rubin smartcard protocol, thus supporting the claim that checking an analysis for guarantee availability, which checks a protocol for goal availability, may help to discover lack of explicitness in the design. However, guarantee availability has a broader impact. Its application to the analysis of the deployed Kerberos protocol ultimately reveals the need for a simple though subtle environmental condition, which the protocol documentation fails to mention, for the protocol to reach one of its primary goals. Upon the analysis of the protocol, our work reveals that strengthening one message is sufficient for the protocol to achieve key confirmation of B to A. By contrast, as we shall detail later, this was previously claimed impossible, by means of the BAN logic, because of intrinsic limitations of the design.

This paper reformulates a previous conference paper in a more detailed yet accessible manner [5], and presents additional case studies. Most importantly, it provides more formal definitions and clarifies the relationship of our principles with the existing design principles. We begin our treatment with a brief account on the limitations of any security claim, due to a clear understanding of the underlying threat model and of the required properties (Sect. 2). Then, we formalize the principle of guarantee availability (Sect. 3), and hint at its past incarnations (Sect. 4). Hence, we outline the method of formal protocol analysis used for our experiments (Sect. 5), and discuss a few significant applications of our principle (Sect. 6). Finally, we derive some conclusions (Sect. 7).

2 Limitations of security claims

At least the last three decades of computer security have taught us that not only are security claims limited to a threat model, but they must also be clearly defined. However, this is perhaps never stated sufficiently.

For example, the mentioned principles of prudent protocol design make no exception and must be interpreted within what was the standard threat model of their times, due to Dolev and Yao [15]. It consists of a powerful spy who is capable of monitoring the entire network traffic creating messages at will. By contrast, honest principals only see their own traffic. Encryption is typically assumed to be perfect, so that no cryptanalysis is possible. For the sake of demonstration, let us consider the popular asymmetric Needham–Schroeder protocol in Fig. 1. It is presented using the standard $A \rightarrow B : m$ notation, whereby A sends a message m to B. Also, message concatenation is denoted by a comma, while fat braces delimit the body of encryption under a key that is written as an index. Looking at the protocol, it is difficult to conclude straight away whether it conforms to the explicitness principle. Lowe's middle-person attack to the protocol confirms that the second message lacks explicitness because it fails to mention the identity of the sender. That identity is "*essential to the meaning of the message*" [2] because its absence causes—within Dolev–Yao's threat model—A's misinterpretation of the originator of the second message and then the attack.

It is interesting to reconsider this argument under different threat models. The original protocol does not seem to lack explicitness within other, more optimistic than Dolev–Yao's, threat models. For example, one is a model where the spy is not a registered principal, hence cannot initiate the protocol. Another one features the spy's inability to interleave protocol sessions, forcing her to engage in only a session at a time. A trivial one sees the spy as a passive eavesdropper. Assuming these models clearly prevents A's misinterpretation about the second message. Therefore, that identity is not "*essential to the meaning of the message*" [2] in such threat models, namely the second message does not lack explicitness, and thus Lowe's attack cannot succeed.

In summary, as with any security statement, establishing whether or not a protocol holds to the explicitness principle is subordinated to the given threat model. The sequel of the paper shall also illustrate how the threat model influences

$$
\begin{aligned}
&1.\ A \rightarrow B : \{\!|Na, A|\!\}_{Kb}\\
&2.\ B \rightarrow A : \{\!|Na, Nb|\!\}_{Ka}\\
&3.\ A \rightarrow B : \{\!|Nb|\!\}_{Kb}
\end{aligned}
$$

Fig. 1 The asymmetric Needham–Schroeder protocol

conformity of a design to goal availability or of an analysis to guarantee availability.

The general guideline given in the opening of this section requires that a security claim be defined without ambiguity. In this vein, Gollmann's account on the meaning of authentication [16] is emblematic. By elaborating on the various understandings of this goal, it demonstrates that also a clear specification of the term "authentication" is relevant to a security analysis. For example, if authentication merely means that a principal is "*indeed present and taking part in the protocol*" [16], then even the original Needham–Schroeder protocol meets authentication of A with B under Dolev–Yao's threat model. This is no longer the case if we take a more stringent and appropriate definition of authentication requiring that "*the responder verifies the source of the challenge it replied to, rather than the source of the reply to its own challenge*" [16].

3 Guarantee availability

Let us begin with the intuition behind the *guarantee availability* principle of realistic protocol analysis.

The design of a protocol is a specification of the protocol steps and of the goals the protocol is expected to achieve. The analysis of a protocol is the evaluation of whether a protocol in fact achieves its expected goals. Goals may typically take either a protocol-centric form—for example, confidentiality: the protocol keeps the session-key from those not intended to know it—or a principal-centric form—for example, confidentiality: the protocol peers know that the session key received through the protocol is kept from those not intended to know it. We address goals of the latter form, where the subject of the goal definition is one or both protocol peers, as *available goals*.

Some goals, such as authentication, are more naturally specified in a principal-centric form that is they are intrinsically meant to be available. (However, this is sometimes forgotten through claims such as: the protocol guarantees authentication of the initiator to the responder. This leaves unspecified what evidence the responder should seek). Some goals have both forms in use, such as key distribution and key confirmation (Sect. 1). Other goals, such as confidentiality, are usually treated only by their protocol-centric form because the principal-centric one is somewhat subsumed: if a protocol keeps a message confidential, then a peer may consider it so from the point in time he gets it.

We argue that all the goals that a protocol aims at should be available (to the protocol peers), which means that all those goals should be achieved in their principal-centric form. We name this principle of protocol design as *goal availability*. Once a new principle is stated, the style of the seminal paper on prudent design principles [2] is to then give examples

of informal analysis to support the new principle. This is meant to provide evidence that adherence to the principle may improve the designs of the protocols. Incidentally, our innovation is to provide that evidence using formal, rather than informal, analysis. However, independently of the use of formal methods, we realize that some guidance is in turn possible and necessary to conduct the analysis with the aim of establishing conformity of a design to goal availability. That guidance is guarantee availability, a principle of protocol analysis.

Once the analysis provides the necessary evidence, the goal availability principle of protocol design can be accepted. We shall see that the byproduct we are left with, the principle of protocol analysis, is most important because it lets us, by itself, deepen the analyses considerably. We first introduce an abstract version of our principle of protocol analysis.

Principle 1 (Guarantee availability—abstract) *The analysis of a protocol should focus on principal-centric goals that is establish whether a protocol achieves available goals, and hence consist of available guarantees.*

The principle states that the protocol analyzer should study available goals by establishing *available guarantees*. The latter are yet to be defined precisely, but it is intuitive to think of guarantees that are established from the principals' viewpoints and are therefore based on facts that the principals can, to a large extent, evaluate by themselves.

An example may help to specify an abstract presentation. Let us consider the toy protocol in Fig. 2. It aims at distributing confidential session keys to its peers with the help of a trusted third principal, the *server* S. Each principal owns a long-term symmetric key and shares it with the server alone. Upon A's request, the server issues a session key Kab and sends it off to both peers within the encrypted tickets,

$$\{\!|A, B, Kab|\!\}_{Kb} \quad \text{and} \quad \{\!|A, B, Kab|\!\}_{Ka},$$

respectively. Admitting a Dolev–Yao attacker, we conduct an informal analysis of the confidentiality goals of the protocol as an attempt to boost the reader's intuition on guarantee availability.

It can be seen that the server sends the session key protected under the principals' long-term keys. It is easy to informally convince ourselves that *if the principals A and B keep their long-term keys from the spy, and the server sends a session key Kab within the tickets* $\{\!|A, B, Kab|\!\}_{Kb}$ *and* $\{\!|A, B, Kab|\!\}_{Ka}$*, then that session key remains confidential*. There is an implicit assumption that the server is secure, that is, it cannot be compromised. However, guarantee availability tells us that if this is as far as our confidentiality guarantee can go, then it is of no use to A, for example. When she receives the ticket $\{\!|A, B, Kab|\!\}_{Ka}$ from the network, she cannot appeal to the guarantee stated above in italic because, even after the ticket inspection, nothing is telling her that it was the server who sent that very ticket. In general, never during her participation in a protocol can A verify what is happening at other ends of the network. In particular, A cannot witness that the server sent her the ticket she just received, which means that she cannot verify that the second precondition of our informal guarantee of confidentiality holds. Therefore, that guarantee is not available to A as it stands.

$$\begin{aligned} &1.\ A \rightarrow \mathsf{S} : A, B \\ &2.\ \mathsf{S} \rightarrow B : \{\!|A, B, Kab|\!\}_{Kb} \\ &3.\ \mathsf{S} \rightarrow A : \{\!|A, B, Kab|\!\}_{Ka} \end{aligned}$$

Fig. 2 A toy protocol

Another guarantee may help: *if A keeps her long-term key from the spy, and receives the ticket* $\{\!|A, B, Kab|\!\}_{Ka}$*, then it was the server who sent her that ticket*. In fact, the first precondition of this guarantee ensures that no one but A or the server can use A's long-term key. By composing the two guarantees just given in italic, a guarantee available to A unfolds: *if the principals A and B keep their long-term keys from the spy, and A receives the ticket* $\{\!|A, B, Kab|\!\}_{Ka}$*, then the session key Kab is confidential*. Principal A can easily check the second precondition of this guarantee—the first precondition is discussed later. It means that A is entitled to consider the session key confidential as soon as she receives a message that has a specific form and contains the key. By contrast, the first guarantee only has theoretical importance if considered from A's viewpoint: it is in fact stated for the server. In conclusion, our analysis has reached a guarantee available to A of session key confidentiality, meaning that the protocol makes session key confidentiality available to A. A homologous guarantee can be developed for principal B.

Having now provided the necessary intuition, and observed the role that the threat model plays, we can refine Principle 1 accordingly.

Principle 2 (Guarantee availability—refined) *The analysis of a protocol should consist of guarantees that are available to a protocol peer within a realistic threat model.*

The main requirement that Principle 2 sets signifies that a guarantee should only rely on assumptions that a peer is able to verify, otherwise he cannot apply it. If none of the existing guarantees about a goal for a peer meets such a requirement, then we say that "the analysis cannot make the guarantees about the goal available to the peer", or that "the analysis fails to conform to the principle of guarantee availability for the given guarantees", or similar statements. We then conclude that "the protocol fails to achieve availability of that goal for that peer" or that "the protocol fails to make that goal available to that peer". As we shall see, such a conclusion bears

vast potential for the peer to be attacked. However, other guarantees about the same goal may exist and be available to the other peer. Also, the principle is strongly dependent on the chosen threat model, so that some guarantee might be available to a peer within a model though unavailable to the same peer within a more realistic model.

It seems didactic to demonstrate here how guarantee availability can confirm protocol insights that we already know by intuition—more significant applications obtaining unknown insights will be discussed (Sect. 6). Let us consider the flawed variant of the toy protocol of Fig. 2 such that the third message is not encrypted. Even informal reasoning easily concludes that *if the principal B keeps his long-term key from the spy, and the server sends a session key Kab within the ticket* $\{\!|A, B, Kab|\!\}_{Kb}$ *but does not send the third message, then that session key remains confidential*. Guarantee availability warns us that we must not be content with such a guarantee of confidentiality because A cannot apply it: A cannot make sure that the server does not send the third message. (In practice, satisfying that precondition would even prevent key distribution). It is impossible to weaken the preconditions of the guarantee and still maintain its conclusion, because the precondition that the server sends the ticket serves to pinpoint the session key. So, we have come to the strongest confidentiality guarantee even for A, namely the confidentiality guarantee cannot be developed any further. In consequence, this analysis cannot make the guarantees of session key confidentiality available to A, and this indicates that the variant toy protocol cannot make session key confidentiality available to A. An analogous negative conclusion can be reached for B: neither B can ever make sure that the server does not send the third message. Therefore, in prescribing analysis from each principal's viewpoint, guarantee availability also guides (even naive) analyzers by helping with the interpretation of the guarantees.

Before we complete the presentation of the principle by defining the concept of *available guarantee* more precisely, it is useful to reconsider the first precondition of the confidentiality guarantee for A on the toy protocol of Fig. 2. Never can A verify that her peer is not disclosing his own long-term key at the other end of the network. Hence, even the final guarantee demands from A the indispensable, minimal trust that its first precondition holds. Also, in the following, a protocol model is generically referred to as an abstract description of the protocol, its goals and the assumed threats.

Definition 1 (*Available guarantee*) Let **P** be a security protocol, $\mathcal{P}$ be a model for **P**, and A be a principal's name. A guarantee in $\mathcal{P}$ is *available to A in* $\mathcal{P}$ if it is established on the basis of assumptions that A is able to verify in $\mathcal{P}$ within her minimal trust.

The network events that an honest principal can realistically verify in a typical protocol model are exclusively whether he sent a specific message to another principal, or whether he received some other specific message from some principal. For example, by Definition 1, a guarantee relying on some principal's sending or receiving a session key should not be considered available to any other honest principal in a protocol model comprising Dolev–Yao's threat model, because this assumes that the spy is the only principal who can monitor the entire traffic.

Detailed protocol models also describe a number of facts that pertain to the environment in which the protocols are executed and that the principals can never verify. For example, no principal can verify that his and his peer's workstations are secure from trojan horse attacks that would disclose all their secrets. Likewise, no principal can make sure that his remote peer keeps from colluding with intruders or that a random point-to-point connection is not broken forever. Therefore, these facts form what we name the principal's *minimal trust*, something that an honest principal can only take for granted. In other words, because no honest principal has complete control over the environment except for his own incoming/outgoing traffic, he can only make assumptions about the rest, which in fact form his trust. If the minimal trust turned out not to hold, then no principal would benefit from engaging in any classical security protocol—with non-repudiation protocols, a principal's minimal trust does not include his peer's honesty. Note that Definition 1 tolerates the minimal trust within an available guarantee.

Definition 2 (*Minimal trust*) Let **P** be a security protocol, $\mathcal{P}$ be a model for **P**, and A be a principal's name. The *minimal trust of A* is the set of environmental facts described in $\mathcal{P}$ whose truth values A needs to know but can never verify in practice.

In brief, Definition 2 calls minimal trust of a principal the set of environmental facts whose truth values the principal needs to establish in order to enjoy the protocol guarantees, although he can only establish them on the sheer basis of his trust because he cannot get adequate evidence. Modulo the minimal trust, the assumptions of a guarantee that we require a principal to be able to verify for the guarantee to be available remain only those concerning the protocol messages. It is known that the form of the protocol messages fundamentally influences the strengths and weaknesses of the entire protocol: the analyzer's focus is directed toward them.

In this paper, we adopt a threat model based on Dolev–Yao's where, in addition, principals' collusion with the spy is explicitly admitted. Hence, in our model, a principal's minimal trust comprises that his peer is neither the spy nor a spy's accomplice, and that encryption is perfect. The minimal trust may be extended in case-specific protocols, such as those based on smartcards, require extensions to the threat model itself. In general, a large minimal trust may characterize a detailed model, while less realistic models, such as one

that precludes collusion with the spy, often take the minimal trust for granted and omit it.

Let us briefly summarize. Guarantee availability tells us that the protocol guarantees must be studied from the principals' viewpoints to check if the peers can apply them within their minimal trust. We shall see how tuning protocol analyses with guarantee availability obtain for us novel protocol insights, and ultimately inspire significant updates to the protocols.

4 Past incarnations of guarantee availability

Although guarantee availability has never been formalized into a principle thus far, it is certain to have influenced some previous works, perhaps implicitly. Here, we advance some comments that are necessarily incomplete due to the large number of formal methods developed in the last decade. As a general remark, the fact that a protocol is specified from each principal's standpoint does not necessarily imply that any guarantees about that protocol model comply with guarantee availability. Such compliance entirely depends on the very statement of the guarantee, which can be influenced by the proof technique in use. Therefore, the analyzer must continuously bear in mind this principle while his work unfolds.

For example, an inductive protocol specification of Paulson is a set of rules, each expressing a fragment of a specific principal's operation. However, his initial proof experiments by induction and theorem proving had to primarily focus on developing proof strategies for the confidentiality guarantee, leaving less consideration for the theorem assumptions [25]. Some of the early proof releases unsurprisingly featured assumptions about the responder that were hardly justifiable from the initiator's viewpoint [35]. Each honest principal's view of the network traffic clearly is limited to what he alone sends or receives. Only later, in studying the authenticity of the cipher texts exchanged in symmetric cryptography protocols did he point out that "agents need guarantees (subject to conditions they can check) confirming that their certificates are authentic" [26], which exactly sounds as a call for guarantee availability. After spelling out the new principle fully, we checked all existing inductive statements against it and therefore had to upgrade a number of statements and corresponding proofs. It is fair to state that the current proof release [34] only features guarantees that are compliant with guarantee availability unless otherwise stated. The upgrade process brought forward some interesting insights that form the examples discussed in this paper.

The compact analysis by BAN logic published by Burrows et al. [13] indeed is conducted from each principal's viewpoint and accurately weighs up the preconditions of each formal statement. This is exactly in the spirit of guarantee availability. For example, their work points out what seem to be inexplicable assumptions for certain proofs to proceed: the analysis of the symmetric cryptography Needham–Schroeder protocol [13] reveals that it is "unusual" and "dubious" that the responder B can merely assume that the received session key is fresh. The same does not apply to the protocol initiator. Because B cannot verify that assumption, we conclude that the analysis fails to make the guarantees of freshness of the session key available to B, that is, the protocol fails the corresponding goal. In consequence, (it is well known that) an intruder can mount a replay attack on the session key against B [14].

The importance of the principals' viewpoints appears to be implicit in the analyses by CSP and model checking. For example, Lowe's influential contribution was initially oriented to only finding attacks [20,21]. Only later was the method tailored to proving protocol goals, and it was natural to define the *process* formalizing each principal's sending *signals* solely on those *channels* that the principal can access. This process-centric style of specifying the protocol models implicitly shows sensitivity to guarantee availability, although our principle pertains to the actual guarantees. It would seem that the proof methods that were subsequently adopted, such as equivalence checking and trace equivalence, inherently comply with guarantee availability [28], but a deeper investigation, currently beyond our aims, appears worthwhile.

The principle of guarantee availability appears to be naturally embedded in Thayer et al.'s [33] analyses by strand spaces. That method lets the analyzer study protocol properties over a *bundle*, which is exactly a collection of actions performed by a single peer. Also, recent advances [3] obtained using the provable security approach of Bellare and Rogaway [11] seem to proceed in the direction of guarantee availability: principals are divided into clients and servers and the security properties are defined with respect to each. Approaches whose proofs of correctness are based on equivalence checking, such as the Spi-calculus [1], only implicitly embed each principal's viewpoint in purely declarative property specifications. However, adherence to guarantee availability must be explicitly studied to gain unknown protocol insights, as we shall see later.

An anonymous referee of this manuscript suggested that, at the level of design, goal availability generalizes the notion of *fail-stop* protocol by Gong and Syverson [17]. A fail-stop protocol is such that an attacker's active tampering with a protocol message causes the protocol to later halt without compromising confidentiality of sensitive components. Therefore, the analysis of fail-stop protocols can safely reduce to passive attackers only. We appreciate some similarity in spirit between the fail-stop property and goal availability limited to the confidentiality goal, although their applications slightly differ. Our principle boils down to specific features of the analysis, which can in turn be guided

by guarantee availability. The notion of fail-stop demands some rigorous preprocessing before the analysis to reduce its scope.

5 The inductive method of protocol analysis

The inductive method of protocol analysis was invented by Paulson in 1996 [25] and developed with the author of the present paper ever since [7]. As its name says, the peculiarity of the method is a thorough use of inductive techniques to specify the protocols, and of the related inductive principles to prove the protocol properties. However, also other methods appeal to induction to prove properties that a protocol preserves [22]. For example, this practice is common to Meadows's NRL Protocol Analyzer to limit the state space for exhaustive search [23] and to Schneider's Rankanalyzer to treat confidentiality [18].

The analysis of a protocol using the inductive method is developed using the proof assistant Isabelle [24] and therefore can be executed mechanically once the human analyzer has finalized it. Isabelle can be freely downloaded from the internet with its repository of proofs [34]. Generality and mechanical support perhaps are the major strengths of the method, which has been applied successfully to a variety of protocols, ranging from deployed ones such as TLS [27], to e-commerce ones such as SET [10], to non-repudiation ones such as the Zhou–Gollmann protocol [9].

Here, there is only room for a brief overview of the method. An unlimited population of principals is defined by a datatype agent over the natural numbers. The datatype includes a *spy*, who monitors the entire network traffic and in consequence knows who sends and who receives which messages. It implements the Dolev–Yao threat model described earlier (Sect. 2). There also exists an unspecified set of *compromised* principals who have colluded with the spy by revealing their long-term secrets, while the remaining principals are *honest*. We shall see that honesty of the peers is a minimal precondition to most guarantees for them. A *server* also belongs to the datatype of principals: it formalizes the trusted third party that many protocols require.

Another datatype intuitively formalizes the *events* representing the protocol execution. Two are the main events. One,

$$\mathsf{Says}\ A\ B\ m$$

signifies that principal A sends message m to principal B. Another one,

$$\mathsf{Gets}\ B\ m$$

expresses B's reception of m. No bound is set on the number of events that each principal can trigger, so that any principal, including the spy, is entitled to initiate at will an unlimited number of sessions of the protocol that is being specified.

The network traffic exactly develops according to the events performed by the principals while they are executing the given protocol. A history of the network is represented by a *trace*, the list of events occurred throughout that history. The set of all possible traces is the formal model for the given protocol and is defined inductively by specific rules drawn from the protocol. For example, if the protocol prescribes that B sends A a message m' upon reception of m, then the model features a rule that may extend a generic trace by the event $\mathsf{Says}\ B\ A\ m'$ each time the trace contains the event $\mathsf{Gets}\ B\ m$. Therefore, the events occur via the firing of the inductive rules. But, as prescribed by induction, no rule is forced to fire, so no event is forced to occur. An extra rule is added to each protocol model in order to formalize the spy's participation in the protocol.

Our binary function knows formalizes the knowledge that principals derive from observing a trace [7]. So, $\mathsf{knows}\ A\ evs$ is the set of messages that principal A either sends or receives on trace evs. Should A be the spy, the set would include all messages that anyone ever sends or receives on the trace. The unary function parts extracts all components (portions of clear-text messages and bodies of cipher texts) from a set of messages; analz is the same but only opens those cipher texts whose encrypting key is available. So, for any set H, we always have that $\mathsf{analz}\ H \subseteq \mathsf{parts}\ H$. Confidentiality of a message component m in a trace evs is expressed as

$$m \notin \mathsf{analz}(\mathsf{knows}\ \mathsf{Spy}\ evs)$$

whereas appearance of m in the network traffic over trace evs is expressed as

$$m \in \mathsf{parts}(\mathsf{knows}\ \mathsf{Spy}\ evs).$$

Principals' long-term symmetric keys that are shared with the server are indicated by function shrK. Likewise, public keys are denoted by function pubK. The language is strongly typed, so that for example a cryptographic message $\{\!|A, B, Kab|\!\}_{Ka}$ is formalized as

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

The event whereby the server sends that message to A is formalized as

$$\mathsf{Says}\ \mathsf{S}\ A\ (\mathsf{Crypt}(\mathsf{shrK}\ A)\{\!|\mathsf{Agent}\ A, \mathsf{Agent}\ B, \mathsf{Key}\ Kab|\!\}).$$

The formal guarantees, which are the focus of guarantee availability, are theorems expressed over a generic trace and are developed under Isabelle's assistance. A typical inductive proof involves checking that each rule—both for base and for inductive steps—of an inductive protocol model preserves a given conjecture. The rule formalizing the spy's operation often requires a dedicated treatment by a specialized proof tactic.

6 Applying guarantee availability

We reviewed in the light of guarantee availability all existing protocol analyses conducted using the inductive method. This section discusses our main findings about four protocols: asymmetric Needham–Schroeder (§6.1), Shoup–Rubin (§6.2), Kerberos (§6.3) and Otway–Rees (§6.4). Applying guarantee availability in these cases always strengthened the analyses and ultimately suggested modifications to the protocol designs.

Also, the very first finding deserves being outlined here: the analysis of the symmetric Needham–Schroeder protocol was strengthened without impact on the protocol design. Precisely, its guarantee of authentication of the initiator with the responder, named **B_trusts_NS5**, as published in earlier releases of Isabelle up to Isabelle98-1 [35], assumes among other facts that the responder sends an instance of the last-but-one message of the protocol. Applying our principle of analysis, we found this assumption to be entirely superfluous to the proof of the theorem, and therefore removed it. The updated version of the theorem can be inspected from the current release of Isabelle [34].

This first finding is enlightening for the practical implications of guarantee availability. Because the guarantee continued to hold with just one assumption fewer, the process inspired no modifications to the protocol design. By contrast, when the removal of the assumptions targeted by guarantee availability falsifies a guarantee, it will inspire modifications to the protocol design in order to enforce the same guarantee despite the missing assumptions. The applications described in the sequel of this section demonstrate this.

To keep the focus on the main argument being accomplished, this paper omits the complete definitions of the protocol models, sometimes simplifies the theorem statements and purposely only sketches the proof details at the end of each subsection—the complete treatment can be found in full elsewhere [7] and is freely released with Isabelle [34]. Also, this paper always specifies which file in the release the treatment comes from.

6.1 The asymmetric Needham–Schroeder protocol

Our initial aim was to investigate the relationship between analyzing protocols in the light of guarantee availability, that is, designing them according to goal availability on one hand, and designing protocols in the light of explicitness on the other hand. We reconsider the analysis of the asymmetric cryptography Needham–Schroeder protocol, shown in Fig. 1 above (§2), and found a clear connection.

Both the original version, subject to Lowe's attack, and the version fixed by Lowe were studied inductively by Paulson within Dolev–Yao's threat model [34]. Let the constant ns denotes the model for the original, flawed protocol. Its inductive definition is omitted here. Lowe's attack against B proceeds from Nb's disclosure to the spy, and hence it is interesting to study whether the guarantee of Nb confidentiality can be made available to B. A relevant guarantee that can be proved is Theorem 1: it is the strongest because its preconditions cannot be relaxed without compromising the conclusion. It comes with file NS_Public_Bad.thy of the latest release of proofs [34].

Theorem 1 (Spy_not_see_NB) *If A and B are honest and evs, belonging to* ns*, contains*

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

but does not contain, for any C,

$$\mathsf{Says}\ A\ C\ (\mathsf{Crypt}(\mathsf{pubK}\ C)(\mathsf{Nonce}\ Nb))$$

then evs is such that $\mathsf{Nonce}\ Nb \notin \mathsf{analz}(\mathsf{knows}\ \mathsf{Spy}\ evs)$. □

While honesty of the peers belongs to B's minimal trust, A's refraining from sending any instance of the second message certainly does not. By definition 1, the guarantee of Nb confidentiality is not available to B, who can never check his peer's activity. Intuitively, non-availability of the guarantee signifies that the existing guarantees of Nb confidentiality are of no use to B. Not surprisingly, when the assumption about A's activity is not verified in practice, Nb indeed loses confidentiality, hence the spy can mount the known man-in-the-middle attack.

Lowe's corrected protocol mentions B in the second message. The analogous reasoning for this protocol, modeled by the constant nsl, is in Theorem 2. It comes with file NS_Public.thy of the latest release of proofs [34]. It is not problematic that theorems 1 and 2 have the same name because they lie in different files. Neither is it accidental: they express the same guarantee for two protocol variants.

Theorem 2 (Spy_not_see_NB) *If A and B are honest and evs, belonging to* nsl*, contains*

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

then evs is such that $\mathsf{Nonce}\ Nb \notin \mathsf{analz}(\mathsf{knows}\ \mathsf{Spy}\ evs)$. □

This theorem signifies that the analysis of the corrected protocol makes a guarantee of Nb confidentiality available to B, because he can check all its assumptions within his minimal trust. Then, B is guaranteed within his minimal trust that his nonce is kept secret (hence, Lowe's attack cannot succeed).

The details of these protocols were known before the present research. However, we are confident that studying availability of the confidentiality guarantee to B would have helped to discover the lack of explicitness in the second message. Faced with Theorem 1, the human analyzer would

have pragmatically checked if any extra explicitness reduced B's assumptions either to verifiable ones or to minimal trust assumptions. Without guarantee availability, it is less easy to evaluate whether Theorem 1 conveys a satisfying guarantee.

To advance just a few proof details, it is interesting to notice that both theorems can be proved routinely. The main assumptions, which concern the Says events, are first inserted in the inductive formula by appealing to the rule of reverse modus ponens. Theorem 1 features two such preconditions and so requires two such appeals, while Theorem 2 can do with one. Then, the classical applications of induction and simplification follow. Isabelle now leaves two subgoals for the human analyzer to solve. One can be terminated routinely by application of the specialized tactic addressing the spy's operation. The last one requires a lemma that honest users generate fresh nonces.

6.2 The Shoup–Rubin protocol

We already published an inductive analysis of the Shoup–Rubin smartcard protocol [6]. We look back at that analysis with the aim of gaining extra insights using our guarantee availability principle. Shoup–Rubin, which is named after its authors, is 11 complex messages long and aims at session key confirmation [19,30]. Each principal stores his own long-term secrets in a smartcard, hence there is no risk of human leak of such sensitive information, even under corruption. Each card can compute a session key using a public algorithm and implements basic encryption/decryption functionalities.

For the sake of formal analysis, a new event denoting card output is defined, Outputs (Card A) $A\,m$, and another for card input, omitted here. Following the protocol authors' threat model, we extend Dolev–Yao's as follows: any principal can lose session keys to the spy by accident, and the spy can use all smartcards unless they are *safe*—a simplification for this paper of the actual sets of *stolen* and of *cloned* cards [6]. Absence of accidents and card safety clearly belongs to any honest principal's minimal trust. We address this as the *original threat model*. Moreover, to investigate how the protocol messages are linked among themselves, we additionally assume that all cards are compromised so that they may omit some outputs, repeat others, and schedule all outputs randomly, namely independently from the input schedule. We address this as the *extended threat model*. It can be naturally embedded in the inductive definition of the formal protocol model, where each inductive rule may fire indefinitely at any time [6].

Phase IV of the protocol is in Fig. 3. It sees B query his card C_b to obtain a session key for use with A. Precisely, B inputs his peer's name and a nonce Na previously obtained from the network. The nonce cannot be used to identify A because it traversed the network in the clear. The card outputs a fresh nonce Nb, the session key, and two tickets

$$6.\ B \rightarrow C_b : A, Na$$
$$7.\ C_b \rightarrow B : Nb, Kab, \{\!|Na, Nb|\!\}_{\pi_{ab}}, \{\!|Nb|\!\}_{\pi_{ab}}$$

Fig. 3 Phase IV of the Shoup–Rubin protocol

that bind the two nonces together. They are encrypted under a key π_{ab}, whose history we omit for brevity, which is only known to A and B's smartcards, not to the principals—this will be important below. Encryption is symmetric. Our model formalizes π_{ab} as pairK(A, B). We remark that the protocol itself prescribes that principals know no long-term keys, which substantiates the crucial role played by the cards. Also, the protocol assumes that the means between each principal and his card are secure in terms of confidentiality and authentication. Therefore, messages 6 and 7 are unintelligible to the spy.

Let the constant sr denotes the formal model for the protocol. Its inductive definition is omitted here. In studying the principal-centric version of session key confidentiality, we find out that the strongest relevant guarantee is Theorem 3: it is impossible to weaken its assumptions without compromising its conclusion. Incidentally, proving this theorem is computationally expensive because all 11 messages must be verified to preserve confidentiality. It comes with file ShoupRubin.thy of the latest release of proofs [34].

Theorem 3 (Confidentiality_B) *If A and B are honest, their both cards are safe, Kab is not leaked by accident, and evs, belonging to* sr*, contains*

$$\text{Outputs (Card } B)\ B\ \{\!|\text{Nonce } Nb, \text{Key } Kab, Cert, \text{Crypt(pairK}(A, B))(\text{Nonce } Nb)|\!\}$$

then evs is such that Key $Kab \notin$ analz(knows Spy evs). □

Does this theorem qualify as a guarantee available to B of session key confidentiality? The assumptions that are stated in prose clearly belong to B's minimal trust. The point here is whether B can verify the main assumption of the theorem, namely that his card outputs a message of a specific form. Although B can verify that his card sends him an output, he cannot verify the form of the fourth component of the output because this is encrypted under a key, pairK(A, B), that he ignores. In consequence, the theorem is not available to B in the extended threat model even within his minimal trust.

To design an equivalent guarantee that is available to B in the extended threat model, inspection of the fourth component should be unnecessary. We realize that doing so hides the identity of B's peer, who instead must be explicit because she needs to be assumed not to be the spy and her card needs to be assumed safe for the session key to remain confidential. Hence, we realize that keeping the conclusion of the theorem without checking the fourth component of the output requires:

– either assuming everyone (thus including B's peer, whoever he might be) not to be the spy and his card to be safe;
– or making B's peer explicit in the output so that the peer alone can be assumed not to be the spy and his card alone to be safe.

The guarantee resulting from the first strategy would still be unavailable to B because he would be unable to check all its assumptions within his minimal trust. As we realize that the guarantee resulting from the second strategy would instead be available to B, it can be appreciated how the check for guarantee availability serves as a check for explicitness also with this protocol. Our reasoning therefore indicates that message 7 fails to tell B the peer with whom the session key is to be used, and hence lacks explicitness in the extended threat model. We realize that this affects also key confirmation of A to B: Theorem **B_keydist_to_A** [34], omitted here, concludes that A knows the session key upon similar assumptions to those of Theorem 3. It means that the guarantee of key confirmation of A to B is not available to B in the extended threat model. Thus, an attack becomes possible whereby B associates the session key with a wrong peer (who, in B's minimal trust, is not the spy and has a safe card) and ends up with a non-confidential session key because the real peer is either the spy or his card is not safe. This is both a key confirmation and a confidentiality attack.

In consequence, we update the protocol so that A's identity is mentioned as second component of message 7. The resulting protocol is modeled by srb. The strongest confidentiality guarantee about it is Theorem 4. It comes with file ShoupRubinBella.thy of the latest release of proofs [34]. The same remark made in the previous section applies to Theorems 3 and 4: they bear the same name as they are expressing the same guarantee in two different files, respectively, about two protocol variants.

Theorem 4 (Confidentiality_B) *If A and B are honest, both their cards are safe, Kab is not leaked by accident, and evs, belonging to* srb*, contains*

$$\mathsf{Outputs}\ (\mathsf{Card}\ B)\ B\ \{|\mathsf{Nonce}\ Nb, \mathsf{Agent}\ A, \mathsf{Key}\ Kab, Cert1, Cert2|\}$$

then evs is such that $\mathsf{Key}\ Kab \notin \mathsf{analz}(\mathsf{knows}\ \mathsf{Spy}\ evs)$. □

Notice that Theorem 4 no longer needs to inspect the form of the fourth component and features no assumptions beyond B's minimal trust. It therefore qualifies as a guarantee that is available to B because its assumptions are either verifiable or contained in the minimal trust. In consequence, the analysis of the upgraded protocol makes a guarantee of session key confidentiality available to B in the extended threat model. The same can be said of key confirmation of A to B, thanks to the updated version of Theorem **B_keydist_to_A** [34]. An equivalent reasoning can be conducted from A's viewpoint.

It was only after our protocol modifications inspired by the analysis in terms of guarantee availability that we realized that Theorem 3 is in fact available to B in the original threat model—this observation is not even published in the full analysis [6]. In that model, B may safely associate the session key received by his card, without inspecting the fourth component of the output, to the last principal name he input the card with. Once more, the relativeness of our claims to the assumed threat model (§2) is demonstrated.

The two theorems discussed in this section succumb to similar proofs. However, the custom steps of each proof terminate with the preparation of the inductive formula and the application of induction. Simplification in fact requires two specific rewriting rules for the analz operator that are easier to express as formulae than in prose. One says that if a cryptographic key can be analyzed from the union of a set of messages and another key, then either the two keys match or the first key can be analyzed from the set of messages alone. Another one states that if a message component can be analyzed from a set of messages, then expanding the set of messages with that component will not make any difference in terms of what can be analyzed. A number of applications of three lemmas are indispensable before and after simplification. They correspond to the three smartcard outputs that this protocol prescribes. Whenever portions of these outputs are implicit, such as *Cert*, *Cert1* or *Cert2* from the Theorem preconditions, applying the lemmas can open them up and express their forms explicitly.

6.3 The Kerberos protocol

The considerations presented so far support the claim that non-availability of a confidentiality guarantee ultimately indicates lack of explicitness in the design. However, this claim does not always hold, as confirmed by the following observations about the Kerberos protocol. We already studied Kerberos IV in detail [8] and now reconsider its analysis in the light of our new principle. We will show that no guarantee of session key confidentiality can be made available to the protocol responder in a realistic model, although this is not due to lack of explicitness.

Kerberos is essentially a key distribution protocol. We outline its main details here. The protocol relies on the Kerberos system, a dual server containing the Kerberos Authentication Server (Kas in short), and the Ticket Granting Server (Tgs in short). While Kas is responsible for authenticating the users to the system, Tgs serves to authorize authenticated users' access to kerberised services. Each principal owns a long-term key, and his minimal trust says that the key is shared with Kas.

1. $A \rightarrow \mathsf{Kas} : A, \mathsf{Tgs}, T1$
2. $\mathsf{Kas} \rightarrow A : \{\!|authK, \mathsf{Tgs}, Ta, \{\!|A, \mathsf{Tgs}, authK, Ta|\!\}_{Ktgs}|\!\}_{Ka}$
3. $A \rightarrow \mathsf{Tgs} : \{\!|A, \mathsf{Tgs}, authK, Ta|\!\}_{Ktgs}, \{\!|A, T2|\!\}_{authK}, B$
4. $\mathsf{Tgs} \rightarrow A : \{\!|servK, B, Ts, \{\!|A, B, servK, Ts|\!\}_{Kb}|\!\}_{authK}$
5. $A \rightarrow B : \{\!|A, B, servK, Ts|\!\}_{Kb}, \{\!|A, T3|\!\}_{servK}$
6. $B \rightarrow A : \{\!|T3 + 1|\!\}_{servK}$

Fig. 4 The Kerberos protocol

The protocol is in Fig. 4. The first two messages form phase I concerning authentication of an initiator to Kas. The third and fourth messages form phase II, concerning authorization from Tgs to the initiator. The last two messages form phase III, concerning the initiator's access to the required service. The *authorization ticket*, $\{\!|A, \mathsf{Tgs}, authK, Ta|\!\}_{Ktgs}$, along with the *authorization key*, *authK*, are delivered to *A* during phase I and serve for phase II. Likewise, the *service ticket*, $\{\!|A, B, servK, Ts|\!\}_{Kb}$, along with the *service key*, *servK*, are delivered to *A* during phase II and serve for phase III.

Notice the timestamp *Ta* associated to *authK*. It has a lifetime of hours, corresponding to the period of time a user can use a workstation session. By contrast, the timestamp *Ts* associated to *servK* has a lifetime of minutes, corresponding to the period of time during which a user can finalize access to a network resource. Other timestamps, *T2* and *T3*, are issued by the senders of messages 3 and 5, respectively, so that a receiver can check that they are fresh messages. Their lifetime is in the order of seconds.

Kerberos relies on timestamps to convey freshness guarantees. From the perspective of formal analysis, this requires extensions to the basic Dolev–Yao threat model in terms of risks for the various session keys. Will they remain confidential forever? A realistic modeling choice was to assume that any session key can be lost to the spy by accident only when its corresponding lifetime expires [8]. Conversely, assuming that no session key can be leaked by accident while it is valid belongs to any principal's minimal trust.

In this threat model, interesting insights originated from studying availability of *servK* confidentiality to the protocol responder. Here, the protocol model is denoted by the constant kerberos. Its inductive definition is omitted from this paper. Theorem 5 is the strongest relevant guarantee. It is based on the homonymous theorem that comes with file KerberosIV.thy of the latest release of proofs [34], but this very version is unpublished. It explicitly refers to the events whereby *B* learns his ticket and the Kerberos servers publish *A*'s. It is more readable than the version published in the release, where the three events are easily transformed into appearance of their three tickets in the traffic.

Theorem 5 (Confidentiality_B_unpublished) *If A and B are honest and evs, belonging to* kerberos*, contains*

Gets B {|Crypt(shrK B){|Agent A, Agent B, Key $servK$, Number Ts|}, X|}

and

Says Tgs A (Crypt $authK${|Key $servK$, Agent B, Number Ts, $servTicket$|})

and

Says Kas A (Crypt(shrK A){|Key $authK$, Agent Tgs, Number Ta, $authTicket$|})

and

¬expiredAK Ta evs *and* ¬expiredSK Ts evs

then evs is such that Key $servK \notin$ analz(knows Spy evs). □

Notice the last two assumptions, which state that both the authorization key and the service key are still valid, namely not yet expired on the given trace. Let us study whether *B* can verify the assumptions of this theorem within his minimal trust. He can check to have received the service ticket containing a timestamp that has not expired, which means he can verify the first and fifth assumptions. Peers' honesty belongs to *B*'s minimal trust. However, *B* cannot verify the remaining three assumptions, which pertain to protocol phases he does not even participate in. Aiming at guarantee availability, we would like to relax these assumptions, which together signify that the authorization key encrypting the service key whose confidentiality is being proved has not expired. At an attempt to relax any of them, the prover leaves us with a subgoal that cannot be proved, hence the theorem is falsified. The subgoal describes the realistic scenario where the authorization key expires is then lost to the spy, and this by mere decryption also leaks the service key.

Because its assumptions cannot be relaxed, Theorem 5 is the strongest guarantee to confirm service key confidentiality to *B*. So, this guarantee is not available to *B* within our threat model. With the experience gained thus far in applying our principle, we observe that an attack against *B* can be built by falsifying exactly those assumptions that the principal cannot verify. In this case, *B* may accept a valid service key that is lost to the spy because it was encrypted under an expired, compromised authorization key. During the lifetime of the service key, *B* will encrypt data he cannot expect will be read by the spy. This is an attack against *B*.

Enforcing guarantee availability in this case does not appear to be a matter of adding explicitness to the protocol. By contrast, because Tgs is a trusted server, it is reasonable to temporally constrain its issuing of service keys as follows:

a service key can be issued only if it is not going to expire later than the authorization key that encrypts it.

From a principal's viewpoint, this is an environmental assumption that must be part of the minimal trust. The upgraded protocol assures that, should an authorization key become lost, the corresponding service keys would be expired. It is then possible to prove a version of Theorem 5 that omits all assumptions that B cannot verify: the two Says assumptions and the expiredAK assumption. Therefore, it qualifies as an available guarantee.

The experiments on this protocol teach us that studying confidentiality availability does not merely help to studying explicitness, but it may also favor unknown protocol insights. Therefore, studying availability of other goals may also be important, as the next section demonstrates.

We terminate also this section with some proof details. The Isabelle release of proofs [34] shows that a lemma of widespread use easily transforms assumptions of the form

trace evs contains event Says *A B X*

or of the form

trace evs contains event Gets *B X*

into X's appearance in the traffic over *evs*, that is

$X \in$ parts(knows Spy *evs*).

Assumptions of this reduced form are the basis of the version of Theorem 5 that is published in the release. Also, the rest of the proof develops in a forward style. First, it derives the confidentiality of the authorization key, as all assumptions to apply that guarantee hold, including the fact that it has not yet expired. This rules out the possibility that Tgs's message discloses the service key. Various applications of technical lemmas conclude. The theorem version for the upgraded protocol also appeals to a lemma stating that if a service key has not expired, then the authorization key that encrypts it has not expired either—clearly this does not hold for the original protocol.

6.4 The Otway–Rees protocol

A version of the Otway–Rees protocol [13], which is named after its authors, appears in Fig. 5. It aims at key distribution using symmetric encryption. As with Kerberos, each principal trusts that his long-term key is shared with the server.

The third message is most important. The server sends B two tickets sealed under A and B's long-term keys, respectively, each containing a copy of the session key. So, B learns the session key from decrypting the second ticket, and forwards the rest to A. Upon reception of the last message, A learns the session key and binds it to B having seen Na returned, which she initially associated to B. Similarly, B binds the session key to A having seen Nb returned, which he associated to A in the second message. This might have been an insecure choice because A's identity arrived in the clear from the first message, but the server confirms the association matching the contents of the two cypher texts received in the second message.

In general, a principal can verify the key confirmation goal out-of-band by probing his peer with session key encrypted traffic and evaluating the replies. However, this cannot be considered a proper protocol property as it stands. Analyzing this protocol clearly reveals that no guarantees of key confirmation of A to B can be made available to B because he cannot be assured that A will ever receive the last message over an insecure network. Key confirmation of B to A is more interesting. The strongest relevant guarantee for A, whose conclusion is Key $Kab \in$ analz(knows B *evs*), cannot be proved without assuming that B receives the third message. In fact, the attempt to relax this assumption in accordance with guarantee availability highlights that the spy can intercept the third message, building the fourth on her own and sending it off to A. This scenario, which is depicted in Fig. 6, clearly subverts the conclusion of the guarantee.

Because of an assumption that A cannot check in our threat model within her minimal trust, the key confirmation guarantee is not available to A. In attempting to enforce it, we have to conclude again that extra explicitness would not help. The violation of key confirmation would derive from the spy's acting as B with the third message. This can be prevented if we give the message some extra integrity by appropriately nesting the encryption operations. The propagation of the session key to A is negligibly delayed. This variant, which is even simpler than what we advanced elsewhere [5], can be seen in Fig. 7. The outer encryption uses B's shared key. We will see that this "extra encryption" does obtain "extra security" [2].

We denote the model for this protocol by the constant orb, but omit its inductive definition here. The new form of the third message allows us to track B's participation. The spy cannot masquerade as him because she cannot decrypt the

1. $A \rightarrow B : M, A, B, \{\!|Na, M, A, B|\!\}_{Ka}$
2. $B \rightarrow \mathsf{S} : M, A, B, \{\!|Na, M, A, B, |\!\}_{Ka}, \{\!|Nb, M, A, B|\!\}_{Kb}$
3. $\mathsf{S} \rightarrow B : M, \{\!|Na, Kab|\!\}_{Ka}, \{\!|Nb, Kab|\!\}_{Kb}$
4. $B \rightarrow A : M, \{\!|Na, Kab|\!\}_{Ka}$

Fig. 5 The Otway–Rees protocol

3. $\mathsf{S} \rightarrow B : M, \{\!|Na, Kab|\!\}_{Ka}, \{\!|Nb, Kab|\!\}_{Kb}$ (intercepted)
4′. $\mathsf{Spy} \rightarrow A : M, \{\!|Na, Kab|\!\}_{Ka}$

Fig. 6 Violating key confirmation of B to A

3. $\mathsf{S} \rightarrow B : M, \{\!|\{\!|Na, Kab|\!\}_{Ka}, Nb, Kab|\!\}_{Kb}$

Fig. 7 Our variant to the Otway–Rees protocol

message. More formally, A's ticket is kept confidential until B extracts it and sends the last message. Theorem 6 confirms this. It is clear that assuming the Says event is indispensable to bind the identity of the responder B. This and the following theorems in this section come with file OtwayReesBella.thy of the latest release of proofs [34].

Theorem 6 (analz_hard) *If B is honest and evs, belonging to* orb*, contains*

$$\begin{aligned}&\mathsf{Says}\ A\ B\ \{\!|\mathsf{Nonce}\ M, \mathsf{Agent}\ A, \mathsf{Agent}\ B,\\ &\quad \mathsf{Crypt}(\mathsf{shrK}\ A)\{\!|\mathsf{Nonce}\ Na, \mathsf{Nonce}\ M,\\ &\qquad \mathsf{Agent}\ A, \mathsf{Agent}\ B|\!\}|\!\}\end{aligned}$$

and is such that

$$\begin{aligned}&\mathsf{Crypt}(\mathsf{shrK}\ A)\{\!|\mathsf{Nonce}\ Na, \mathsf{Key}\ Kab|\!\}\\ &\quad \in \mathsf{analz}(\mathsf{knows}\ \mathsf{Spy}\ evs)\end{aligned}$$

then evs contains

$$\begin{aligned}&\mathsf{Says}\ B\ A\ \{\!|\mathsf{Nonce}\ M,\\ &\quad \mathsf{Crypt}(\mathsf{shrK}\ A)\{\!|\mathsf{Nonce}\ Na, \mathsf{Key}\ Kab|\!\}|\!\}.\end{aligned}$$

□

Notice that the theorem fails to hold if we replace analz by parts because the ticket appears as a component of the traffic even before B sends the last message. We emphasis that neither does the theorem hold of the original protocol, where the assumption evaluates to true even before B's participation.

The second portion of the key confirmation guarantee is in Theorem 7. It asserts that an honest principal sends the last message of the protocol only upon reception of a suitable instance of the third message.

Theorem 7 (OR4_imp_Gets) *If A and B are honest and evs, belonging to* orb*, contains*

$$\begin{aligned}&\mathsf{Says}\ B\ A\ \{\!|\mathsf{Nonce}\ M,\\ &\quad \mathsf{Crypt}(\mathsf{shrK}\ A)\{\!|\mathsf{Nonce}\ Na, \mathsf{Key}\ Kab|\!\}|\!\}\end{aligned}$$

then, for some Nb, evs contains

$$\begin{aligned}&\mathsf{Gets}\ B\ \{\!|\mathsf{Nonce}\ M,\\ &\quad \mathsf{Crypt}(\mathsf{shrK}\ B)\{\!|\mathsf{Crypt}(\mathsf{shrK}\ A)\{\!|\mathsf{Nonce}\ Na,\\ &\qquad \mathsf{Key}\ Kab|\!\},\\ &\qquad \mathsf{Nonce}\ Nb, \mathsf{Key}\ Kab|\!\}|\!\}.\end{aligned}$$

□

We now have all fragments for A's guarantee. Upon reception of the last message of the protocol, A concludes that the ticket is lost to the spy, who sees the entire traffic. This means that the second assumption of Theorem 6 holds. By an appeal to that theorem, A derives that B indeed participated. Theorem 7 can now be applied, confirming that B received an intelligible message quoting the same session key received by A. Certainly, B can decrypt a message sealed under his key and extract the session key, as stated formally by a few subsidiary lemmas about analz. This proves the main guarantee, Theorem 8. It formally states that the protocol informs A that the session key she receives is also known to her peer B.

Theorem 8 (A_keydist_to_B) *If A and B are honest and evs, belonging to* orb*, contains*

$$\begin{aligned}&\mathsf{Says}\ A\ B\ \{\!|\mathsf{Nonce}\ M, \mathsf{Agent}\ A, \mathsf{Agent}\ B,\\ &\quad \mathsf{Crypt}(\mathsf{shrK}\ A)\{\!|\mathsf{Nonce}\ Na, \mathsf{Nonce}\ M,\\ &\qquad \mathsf{Agent}\ A, \mathsf{Agent}\ B|\!\}|\!\}\end{aligned}$$

and

$$\mathsf{Gets}\ A\ \{\!|\mathsf{Nonce}\ M, \mathsf{Crypt}(\mathsf{shrK}\ A)\{\!|\mathsf{Nonce}\ Na, \mathsf{Key}\ Kab|\!\}|\!\}$$

then

$$\mathsf{Key}\ Kab \in \mathsf{analz}(\mathsf{knows}\ B\ evs).$$

□

This theorem lets us conclude that our variant of the Otway–Rees protocol makes the goal of key confirmation of B to A available to A. Following this conclusion, we looked back at the analysis of the original protocol by BAN logic, and spotted the comment: "it is interesting to note that this protocol does not make use of *Kab* as an encryption key, *so* neither principal can know whether the key is known to the other" [13]. The claim is that the protocol fails to make the goal of key confirmation available to its peers *because* it does not make use of the session key as an encryption key.

Our variant protocol falsifies this claim. Although it does not use the session key to encrypt messages, it confirms to A that her peer knows it. Therefore, we realize that prescribing no use of a session key after it is delivered must not be taken as generally undermining availability of key confirmation to the second principal who receives the key (the responder in this case).

The proof of Theorem 6 is long. In particular, to terminate the subgoal arising from the last protocol step, it is necessary to simplify the assumption

$$\begin{aligned}&\mathsf{Crypt}(\mathsf{shrK}\ A)\{\!|\mathsf{Agent}\ B, \mathsf{Nonce}\ M, \mathsf{Nonce}\ Na, \mathsf{Key}\ Kab|\!\}\\ &\quad \in \mathsf{analz}(\{\mathsf{Key}\ K\} \cup (\mathsf{knows}\ \mathsf{Spy}\ evs)).\end{aligned}$$

Its symbolic evaluation is not trivial because expanding the traffic over *evs* with the key K may potentially reveal the ticket if this appears anywhere in the traffic protected by K. Fortunately, this is not the case with this protocol, hence a rewriting rule can be proved to inform the simplifier that no key is used to re-encrypt a ticket. So, that assumption can simplify to

$$\begin{aligned}&\mathsf{Crypt}(\mathsf{shrK}\ A)\{\!|\mathsf{Agent}\ B, \mathsf{Nonce}\ M, \mathsf{Nonce}\ Na, \mathsf{Key}\ Kab|\!\}\\ &\quad \in \mathsf{analz}(\mathsf{knows}\ \mathsf{Spy}\ evs).\end{aligned}$$

Theorem 7 can be proved conventionally: insertion of the main assumption in the inductive formula, induction, a few applications of a lemma concluding that all items of sent or received messages appear in the traffic, then simplification and finally classical reasoning. As we already sketched earlier, proving Theorem 8 reduces to a forward style application of the previous results.

7 Conclusions

The most obvious protocol design principle is protocol correctness, namely that a protocol meet its goals. Because this principle as it stands is difficult to apply, researchers conceived the prudent design principles hoping that they would be easier to check and that they would guarantee conformity to the most obvious design principle. However, both hopes turned out to be badly put. On one hand, many design principles remain difficult to check—for example, it remains difficult to establish when "the identity of a principal is essential to the meaning of a message" [2]. On the other hand, it is now understood that the principles are neither necessary nor sufficient to protocol correctness. They are not sufficient because they fail to cover the variety of strategies inspiring modern designs. They are not necessary because the message features as prescribed by some principles counter certain goals—for example, explicitness often counters privacy. The examples discussed in this paper show that goal availability often implies explicitness. However, that is the case only when explicitness is necessary for a protocol to meet its goals, otherwise the implication does not hold. We argue that the same considerations can be made by reading any of the existing design principles [2] in the place of explicitness. In consequence, by being formulated in terms of goal achievement rather than of message features, goal availability might be the sole design principle to use.

Most importantly, being simply the principal centric version of protocol correctness, goal availability has the extremely useful feature that, contrarily to the existing design principles, the analysis in its support is very easy to guide: just attempt to relax all assumptions that a principal cannot realistically check. This is guarantee availability, which originates as the necessary guidance to analyses in support of goal availability. It is not surprising because all principles of protocol design require supporting evidence that is customarily obtained by protocol analysis. Guarantee availability therefore is a principle of realistic protocol analysis, whether this is formal or informal, whether this is machine assisted or not. The human analyzer should establish in a realistic threat model that the protocol goals are achieved on the basis of assumptions that the principals can verify within their minimal trust.

Protocol analyses often end up in strengthened protocol designs. In this vein, we have also shown that analyses conforming to guarantee availability become deeper, and may therefore inspire additional, significant design modifications that would perhaps be neglected otherwise. The protocols will ultimately become more robust. Additionally, guarantee availability is rather mechanical to check, as the analyzer merely needs to evaluate what a principal can or cannot establish to hold in the given threat model and what belongs to the principal's minimal trust. This is possible because it is recognized that a threat model must be specified before conformity to the principle can be claimed.

Guarantee availability has never been spelled out completely thus far although it must have influenced the protocol verification community in the last 20 years—it certainly guided our own inductive treatments [7]. In this paper, we have detailed its application to four emblematic formal analyses we had conducted before, but expect significant outcomes from application to all protocol analyses.

Acknowledgments The research was supported by the Engineering and Physical Sciences Research Council [grant number GR/R01156/R01]. Colin Boyd, Dieter Gollmann, Andy Gordon, Larry Paulson, Peter Ryan and the anonymous referees gave invaluable advice.

References

1. Abadi, M., Gordon, A.: Reasoning about cryptographic protocols in the spi calculus. In: Mazurkiewicz, A.W., Winkowski, J. (eds.) Proceedings of the 8th International Conference on Concurrency Theory (CONCUR'97), LNCS 1243, pp. 59–73. Springer (1997)
2. Abadi, M., Needham, R.M.: Prudent engineering practice for cryptographic protocols. IEEE Trans. Softw. Eng. **22**(1), 6–15 (1996)
3. Abdalla, M., Fouque, P.A., Pointcheval, D.: Password-based authenticated key exchange in the three-party setting. IEE Proc. Inf. Secur. **153**(1), 27–39 (2006)
4. Anderson, R., Needham, R.M.: Robustness principles for public key protocols. In: Coppersmith, D. (ed.) Proceedings of Advances in Cryptography (CRYPTO'95), LNCS 963, pp. 236–247. Springer (1995)
5. Bella, G.: Availability of protocol goals. In: Panda, B. (ed.) Proceedings of the 18th ACM Symposium on Applied Computing (ACM SAC'03), pp. 312–317. ACM Press (2003a)
6. Bella, G.: Inductive verification of smartcard protocols. J. Comput. Secur. **11**(1), 87–132 (2003b)
7. Bella, G.: Formal Correctness of Security Protocols. Information Security and Cryptography. Springer, Berlin (2007)
8. Bella, G., Paulson, L.C.: Kerberos Version IV: inductive analysis of the secrecy goals. In: Quisquater, J.J., Deswarte, Y., Meadows, C., Gollmann, D. (eds.) Proceedings of the 5th European Symposium on Research in Computer Security (ESORICS'98), LNCS 1485, pp. 361–375. Springer (1998)
9. Bella, G., Paulson, L.C.: Accountability protocols: formalized and verified. ACM Trans. Inf. Syst. Secur. **9**(2), 1–24 (2006)
10. Bella, G., Massacci, F., Paulson, L.C.: Verifying the SET registration protocols. IEEE J. Sel. Areas Commun. **21**(1), 77–87 (2003)
11. Bellare, M., Rogaway, P.: Provably secure session key distribution—the three party case. In: Proceedings of the 27th ACM SIGACT Symposium on Theory of Computing (STOC'95), pp. 57–66. ACM Press (1995)

12. Brackin, S.: A HOL extension of GNY for automatically cryptographic protocols. In: Proceedings of the 9th IEEE Computer Security Foundations Workshop (CSFW'96), pp. 62–76. IEEE Press (1996)
13. Burrows, M., Abadi, M., Needham, R.M.: A logic of authentication. Proc. R. Soc. Lond. **426**, 233–271 (1989)
14. Denning, D.E., Sacco, G.M.: Timestamps in key distribution protocols. Commun. ACM **24**(8), 533–536 (1981)
15. Dolev, D., Yao, A.: On the security of public-key protocols. IEEE Trans. Inf. Theory **2**(29), 198–208 (1983)
16. Gollmann, D.: On the verification of cryptographic protocols—a tale of two committees. In: Schneider, S., Ryan, P.Y.A. (eds.) Proceedings of the Workshop on Secure Architectures and Information Flow, ENTCS 32, pp. 42–58. Elsevier (2000)
17. Gong, L., Syverson. P.: Fail-stop protocols: an approach to designing secure protocols. In: Iyer, R.K., Morganti, M., Fuchs, W.K., Gligor, V. (eds.) Proceedings of the 5th International Working Conference on Dependable Computing for Critical Applications (DCCA'95), pp. 79–100. IEEE Press (1998)
18. Heather, J., Schneider, S.: Towards automatic verification of authentication protocols on an unbounded network. In: Proceedings of the 13th IEEE Computer Security Foundations Workshop (CSFW'00), pp. 132–143. IEEE Press (2000)
19. Jerdonek, R., Honeyman, P., Coffman, K., Rees, J., Wheeler, K.: Implementation of a provably secure, smartcard-based key distribution protocol. In: Quisquater, J.J., Schneier, B. (eds.) Proceedings of the 3rd Smartcard Research and Advanced Application Conference (CARDIS'98), pp. 229–235. (1998)
20. Lowe, G.: Breaking and fixing the Needham–Schroeder public-key protocol using CSP and FDR. In: Margaria, T., Steffen, B. (eds.) Proceedings of the 2nd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'96), LNCS 1055, pp. 147–166. Springer, Berlin (1996)
21. Lowe, G., Roscoe, A.W.: Using CSP to detect errors in the TMN protocol. IEEE Trans. Softw. Eng. **3**(10), 659–669 (1997)
22. Meadows, C.: Invariant generation techniques in cryptographic protocol analysis. In: Proceedings of the 13th IEEE Computer Security Foundations Workshop (CSFW'00), pp. 159–169. IEEE Press (2000)
23. Meadows, C.A.: The NRL protocol analyzer: an overview. J. Log. Program. **26**(2), 113–131 (1996)
24. Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL: A Proof Assistant for Higher-Order Logic. Springer, LNCS Tutorial 2283 (2002)
25. Paulson, L.C.: Proving properties of security protocols by induction. In: Proceedings of the 10th IEEE Computer Security Foundations Workshop (CSFW'97), pp. 70–83. IEEE Press (1997)
26. Paulson, L.C.: The inductive approach to verifying cryptographic protocols. J. Comput. Secur. **6**, 85–128 (1998)
27. Paulson, L.C.: Inductive analysis of the internet protocol TLS. ACM Trans. Comput. Syst. Secur. **2**(3), 332–351 (1999)
28. Ryan, P.Y.A., Schneider, S., Goldsmith, M., Lowe, G., Roscoe, A.W.: Modelling and Analysis of Security Protocols. Addison-Wesley (2001)
29. Schneider, S.: Verifying authentication protocols with CSP. In: Proceedings of the 10th IEEE Computer Security Foundations Workshop (CSFW'97), pp. 3–17. IEEE Press (1997)
30. Shoup, V., Rubin, A.: Session key distribution using smartcards. In: Maurer, U. (ed.) Advances in Cryptology (Eurocrypt'96), LNCS 1070, pp. 321–331. Springer (1996)
31. Song, B., Kim, K.: Two-pass authenticated key agreement protocol with key confirmation. In: Roy, B.K., Okamoto, E. (eds.) Proceeings of 1st International Conference in Cryptology in India, Indocrypt 2000, LNCS 1977, pp. 237–249. Springer (2000)
32. Syverson, P.F.: Limitations on design principles for public key protocols. In: Proceedings of the 15th IEEE Symposium on Security and Privacy (SSP'96), pp. 62–72. IEEE Press (1996)
33. Thayer, F.J., Herzog, J.C., Guttman, J.D.: Strand spaces: proving security protocols correct. J. Comput. Secur. **7**, 191–220 (1999)
34. URL (2009a) Isabelle download page. http://www.cl.cam.ac.uk/Research/HVG/Isabelle/download.html
35. URL (2009b) Old Isabelle releases. http://www.cl.cam.ac.uk/Research/HVG/Isabelle/download_past.html