

What is Correctness of Security Protocols?

Giampaolo Bella

Dipartimento di Matematica e Informatica, Università di Catania
Viale A.Doria 6, I-95125 Catania, Italy
`giamp@dmf.unict.it`

Abstract. This title question has been seeing a number of researchers up many nights long. As soon as major protocol flaws were discovered empirically — a good luck that is not older than the early 1990s — this question came up to the world. It was soon realised that some notion of formal correctness was necessary to substantiate the confidence derived from informal analyses. But protocol correctness was born in a decade when security in general was only beginning to ferment.

Security protocols aim at such a variety of goals that only their various human understandings could further enlarge. This is partly due to the increasing domains where the protocols are finding an application, such as secure access to local-area network services, secure e-mail, e-commerce, public-key registration at certification authorities and so on. But it is also significantly due to the variety of interpretations that virtually each researcher tends to use for each goal.

As it is clear to any expert in formal methods, it is impossible to study protocol correctness profitably without a universal and disambiguous interpretation of its goals. What may be typical of security problems is that it is at least as important to state a detailed and appropriate model of threats that a secure system is meant to withstand. This has been a second and significant source of perhaps useless debates around this or that protocol. Only a few, though eminent, accounts are published to clarify these issues.

These are certain to be some of the reasons why dozens of papers appeared about one, now popular, protocol attack in just a few years of the second half of the last decade. One of the protocol designers firmly refused those "findings" because his protocol had been conceived within a different threat model — and perhaps for different goals — from the one that the publications had been constructed upon. Not only does this seem a gross mistake, but it is somewhat surprising that those publications mostly came from experts in formal methods.

It should be obvious to anyone that an ant will resist if covered with a single sheet of paper but will not survive under a hard-back bulky book. Otherwise, it should be clarified what an ant and a bulky book really are. As a researcher in formal methods applied to the verification of security protocols, I pay particular attention to these issues. With this position paper, I intend to discuss some findings of mine in the area. Their significance mostly is methodical for protocol verification rather than specific for particular protocols. I will then outline my favourite tool, the Inductive Method, and conclude with a few open problems that the community of researchers may want to look at.

1 Premise: Security and Security Protocols

It has become easier and easier to introduce the main notions of computer security to anyone. Even computer illiterates such as my old uncle and auntie have the feeling that it is somewhat related to people's capitals and interests in general. This reality supports the claim that computer security currently has vast social impact, and is therefore worth of attention. However, gone are the 1980s, when the US Department of Defense issued the Orange Book, a set of criteria for evaluating the security of computer systems. It is still very relevant though constrained to stand-alone computers: ours is an era of computer networks.

The Orange Book prescribes that a secure system cannot exist without a clear security policy specifying who can or cannot do what in a system. Ever since 1960, the Compatible Time Sharing System developed at MIT introduced a password file storing each login and related password. By security policy, that file must not have been readable by anyone except the administrator. The policy is then to be implemented as a reliable access-control mechanism. The access-control list bundled with each file in a Unix system is an instance of this concept.

It is understood that a user-authentication mechanism is also fundamental, and passwords have been surpassed by smart-tokens and biometric authentication. The Orange Book also requires auditing. At present, it is widely acknowledged that security is impossible without a full audit trail that logs all significant events occurred in a system, such as system calls, accesses to restricted memory areas and so on. Moreover, all of these measures must be continuously in place and well documented.

In a networked world, the threats have only increased. Some attempts exist to overcome the intrinsic limitations of the Orange Book and provide a complete description of computer security in the new setting. However, I feel that none of them has encountered sufficient acceptance, perhaps because it has become impossible to define the complete toolbox indicating security. To say it with Anderson [5], security rather is a complex CNF formula than a simple boolean predicate. And the real length of that formula seems to remain an open problem as more and more applications demand security.

An example of the growing call for security can be easily found in an academic context. At present, my Department manages student exams and final dissertations electronically. Each lecturer has been provided with a smartcard containing his private key, which he uses to digitally sign exams. All web pages are only HTTPS accessible, meaning that the SSL protocol is securing each transaction. The system now rests on all available security measures thanks to the efforts of local researchers and technicians, but it has been through various insecure stages. For example, a dedicated server, called the Uniweb server, managed the final dissertations for some time. Lecturers' authentication was done via passwords specific for Uniweb, but password reminders were managed insecurely. Anyone could open up the URL of Uniweb's interface, enter a valid e-mail address and have a new password sent to that e-mail. Not only could an attacker mount a denial-of-service attack (DoS) on any lecturer by causing continuous password reminders, but he could also attempt to sniff the new password and

then impersonate the lecturer. This was perfectly realistic because the Uniweb server spoke with the e-mail server in the clear. Obviously, when pen and paper were used a couple of years ago, none of these computer security issues existed.

Larger-scale examples are just as easy to derive. The mentioned SSL protocol is most popular. All it does is to distribute session keys to its peers. It has been used for e-commerce with the result of massive frauds being mounted against merchants a few years ago. The attackers impersonated legal cardholders by sending over SSL well-formed credit card fake numbers to the merchants, who were content with just the well-formedness of the sixteen digits and delivered the goods. Nothing in SSL guarantees to a merchant that a credit card number corresponds to a valid account and a guaranteed payment. The protocol leaves it to the merchant to fetch such a validation “out of band”. Secure Electronic Transactions, SET, is a protocol developed by a credit card consortium exactly upon the empirical evidence that security in the e-commerce domain is much more than mere confidentiality of session keys. E-commerce demands delicate security goals such as confirmation of all transaction details from and to all parties, including a bank or its representative. Forms of partial sharing are also important: the cardholder has to share his payment details (how he is paying) with the bank but may want to keep them from the merchant, who, however, must confirm them to the bank. Dually, the cardholder may accept to share the order information (what he is buying) with the merchant but not with the bank, who, however, must confirm that that order is covered by appropriate payment. SET accomplishes these complex goals [14]. The cardholder is protected because the merchant can confirm the payment details to the bank without seeing them. Likewise, the bank can confirm the order information (relative to a payment) to the merchant without seeing it.

These examples are also useful to highlight that a number of measures for network security are necessary at present. No-one forgets the Orange Book, but no-one considers a network reliable without techniques to thwart malicious software, such as antiviruses and antispyware. Measures to detect and handle potential intruder processes, such as intrusion detection and management systems, are also indispensable. And no network will survive without the perimetrical protection provided by a properly configured firewall.

Even with all these measures in good order, a computer network runs the risks deriving from its own definition: the information that is sent across between nodes. I have already mentioned that sensitive information such as a password traversing the network in the clear essentially vanishes everything. Whoever exhibits a password can impersonate the password’s legal owner, and traffic sniffers that capture the network packets are freely available. This brings us on to the necessity of security protocols: any network traffic must be regulated by specific protocols conceived to accomplish certain security goals. It is known that these protocols typically make use of cryptographic measures, so that a ciphertext only is intelligible to whoever has access to the key that was used to encrypt it. For example, the mentioned SSL and SET make an intelligent use of cryptography to reach their aims.

2 Security Protocol Analysis

Security protocols are well-known to be highly prone to errors. Sometimes, the international community had been convinced for more than a decade that a protocol was essentially OK when it was found to hide subtle attacks. The work of Burrows et al. [19], best known as BAN logic, pioneers these problems in 1989. A variety of attacks are published in the 1990s, such as those by Syverson [38] and by Lowe [32] to just cite a couple.

An attack found to a protocol denounces that the protocol is incorrect, namely it fails to live up to the promises that its designers made about it. But what makes security protocols so difficult to get right, so predisposed to incorrectness? A protocol is a distributed program executed concurrently by a virtually infinite population of agents. In practice, that population is extremely large and impossible to delimit. For example, it is unknown how many are executing SSL this very moment. The concurrent nature implies that any agent can open up any number of protocol sessions with anyone at the same time. Sessions can be broken because of network troubles or be quit purposely. They can even be delayed, so that an agent may want to execute a full session with a peer during the lapse between two steps of another session with another agent. This form of session interleaving has often turned out problematic for security.

The formal method community has profoundly contributed to develop this area. The citations I made above pertain to abstract, logical reasoning performed by pen and paper [19, 38], and to process calculi equipped with state-enumeration techniques [31]. Up to our days, the majority of formal method experts have somewhat tried out their favourite methods at protocol analysis with alternate results. These years' proceedings of international conferences focusing on security do not reach their end without some advance in protocol verification [1, 6, 8, 9, 20, 21, 27].

All these contributions assume that encryption is totally reliable, namely that a ciphertext cannot be opened anyhow without its encrypting key. With this assumption, often known as black-box cryptography, researchers neglect all cryptographic details and concentrate on a variety of security goals. Another group of contributions, pioneered by Bellare and Rogaway [17] and developed till now [3], relaxes this assumption by a probabilistic assessment of the robustness of the cryptographic algorithms. The analysis typically concentrates on the confidentiality goal, which is reduced to the negligible probability that an attacker breaks the cryptographic scheme. Both groups have guided significant advances, so that it is impossible to state that one is more important. It often appears that their findings complement each other. However, it is unfortunate that very few attempts [2, 26] exist to conjugate the two groups and fade out what seems to remain a dichotomy. My own contribution assumes black-box cryptography [11].

3 Recent Findings on Protocol Correctness

The vast majority of publications in this area follow a stable pattern: here is a protocol that is worth considering, here is a major attack to it, and here is our

new method to find it. I have often not resisted this pattern, but will resist it this time. Here, I intend to discuss some findings of mine that may have a methodical influence on how security protocols ought to be analysed, and eventually on protocol development and deployment. In particular, I will demonstrate the importance of defining the goals and the threat model precisely (§3.1), which may help when we wonder whether we can live with flawed protocols. I will also advance a principle of reliable protocol analysis, adherence to which may unveil additional protocol insights (§3.2). I will purposely attempt to dispose with the formal overhead as much as possible, and mostly work out on the reader’s intuition.

3.1 Goals and Threats

It is out of question that the most popular protocol is a simple three-message one based on public-key cryptography and developed by Needham and Schroeder in 1978. It has been debated so much that it is slightly embarrassing to quote it again here. But it provides a good workbench to expand my argument.

1. $A \rightarrow B : \{Na, A\}_{Kb}$
2. $B \rightarrow A : \{Na, Nb\}_{Ka}$
3. $A \rightarrow B : \{Nb\}_{Kb}$

Encryption is assumed to be perfectly reliable, so that only who has the private key corresponding to Kb can open up messages 1 and 3. If nothing wrong happened with key management, that must be B . Likewise, only A can decipher message 2. Each agent can use the random number generator to invent his own nonce: Na comes from A , and Nb originates with B . If nonces are assumed truly random, then A concludes that her peer is B as soon as she receives message 2, which quotes her nonce. And B deduces that her peer is A when he gets message 3, which mentions his nonce. Incidentally, message reception is not guaranteed in general over an insecure network, but if any protocol message fails to reach its intended recipient, than the entire protocol fails.

This was a convincing reasoning for everyone until Lowe published an attack to the protocol in 1995 [30].

1. $A \rightarrow C : \{Na, A\}_{Kc}$
- 1'. $C \rightarrow B : \{Na, A\}_{Kb}$
- 2'. $B \rightarrow A : \{Na, Nb\}_{Ka}$
2. $C \rightarrow A : \{Na, Nb\}_{Ka}$
3. $A \rightarrow C : \{Nb\}_{Kc}$
- 3'. $C \rightarrow B : \{Nb\}_{Kb}$

The attack proceeds from A ’s initiating a session with the attacker C . This seems realistic as no-one knows who the attacker is. It can be seen that the attacker interleaves his communication with A to a session, indicated by the primes, with

some B . He fakes message $1'$ by reusing A 's nonce and identity. Then, he cannot decipher B 's reply $2'$ as it is meant for A , but merely forwards it on to A . It can be seen that, because A cannot register any irregularity, she opens up B 's message and reveals nonce Nb to the attacker in message 3. The attacker is now able to terminate his session with B successfully.

It is clear that B terminated a session with the attacker believing it was A : the attacker has impersonated A with B . The protocol is incorrect because authentication of A with B fails in this case. Lowe reports a drastic consequence based on the fact that the nonces are meant to remain confidential. Following the attack, B believes to share secrets Na and Nb with A , when in fact he shares them with C . Therefore, if B were a bank the attacker could require a money transfer from A 's account to his own by exhibiting the two nonces. The bank B would believe the request came from A , and hence second the transaction.

Questioning the goals. If we are to properly evaluate these findings, we must wonder what the real goals of this protocol were, according to its designers. There is no point lamenting that a car does not go at 1000Mph because it only was conceived for 100Mph. What is authentication? It is in fact the most discussed protocol goal, perhaps because it is easy to misunderstand. Lowe and Gollmann published two separate, eminent treatments [25, 33].

Lowe in particular highlights the weakest form of authentication, which he calls *aliveness*. It holds of an agent when he merely is alive, active in the network. A more stringent form of authentication is *weak agreement*. There is weak agreement of A with B if B believes that A is running the protocol with him, and this is true. This hierarchy of authentication specifications has four levels. The next level, *non-injective agreement* subsumes weak agreement by also requiring agreement on some set of message components. The final level, *injective agreement*, requires a perfect match between the peers' sessions, but has not encountered much mention.

With some definitions in hand, it can be concluded that Lowe's scenario does not count as an authentication flaw if authentication means aliveness. When B believes that his peer is A , she is in fact up and running (though not with B). Only if authentication at least means weak agreement is Lowe's scenario an authentication flaw: B gets his peer wrong. However, these arguments still remain to be matched against the protocol designers' stated goals. It happens that their concerns were even more fundamental, as I say below.

It must be noted that Lowe's attack (or "attack"?) is conceived under the well-known Dolev-Yao's threat model [24]. It sees a super-powerful attacker who can intercept all network traffic and break concatenated messages and ciphertexts of which he knows the encrypting key. He can also build fake messages at will out of the components he knows. Dolev-Yao's clearly is the most potent attacker that can be conceived without relaxing the assumption of black-box cryptography.

Questioning the threat model. Roger Needham refused Lowe’s attack not much on the basis of an imprecise definition of authentication but more fundamentally on the basis of a wrong threat model. There is no point observing that a car cannot fly, because its stated threat model is the road, not the sky.

Needham said that the public-key Needham-Schroeder protocol had been conceived for closed environments where the attacker was an outsider, not a registered agent. It was thus impossible to initiate the protocol with him. Clearly, without this prerequisite, Lowe’s attack would not proceed. However, this reply received some criticism: if a trust relationship holds between the peers so that none of them can be the attacker, it is unclear what an authentication protocol is at all needed for.

When I formalised Lowe’s scenario using soft constraint programming [12], I found out something that had never been reported. Because of C ’s activity, B gets to learn nonce Na even if he only acted legally. Is this a problem? Agent A invents that nonce to share it with its intended peer, who is C . Therefore, the designers would agree that Na is only meant to be shared between A and C , namely it should remain confidential for them. It unquestionably follows that B ’s learning Na counts as a protocol flaw.

It was enlightening to evaluate the putative consequences of this finding. Could B ever take advantage of his knowledge of Nb ? An initial negatively-flavoured answer derived from the fact that B is unaware of the power of Na . But no-one would give his house keys to thefts merely because they do not know which house they open. If the motivations to commit the crime are sufficient, the attacker will try out all keys in his key ring virtually everywhere. This is particularly feasible if there is some proximity relationship between the attacker and the potential victims. It is the case of Lowe’s scenario for example, where all peers are registered for the same protocol.

Having left that debate behind, I continued my investigation on the universally accepted security principle that any violation of a stated goal will be sooner or later exploited by an attacker. But I realised that formally B could not be the attacker because that was C . It was then when I started to question the appropriateness of Dolev-Yao’s model, and soon came to a negative conclusion. That model had been formalised in the early 1980s, when a computer network was a resource limited to a few worldwide institutions and no-one else. It was perfectly realistic to assume that when two representatives of the same institution meant to communicate remotely, they would trust each other and not the rest of the world. Dolev and Yao appropriately formalised the worst case in which the rest of the world would collude against that single line of communication.

It is difficult to picture this scenario nowadays because it just is no longer adherent. Each house potentially is an Internet user, and offensive resources have become cheap both in terms of hardware/software costs and in terms of skill. The Internet even stores freely accessible databases of malicious code for teenagers to try out for fun. The news are continuously reporting about new frauds being mounted within the most unsuspected contexts. I thus envisaged and defined a new threat model, where each agent can either be one who does

not break the rules, a *good*, or a Dolev-Yao's attacker, a *bad*, or one with no specific commitment, an *ugly* [13]. This is the *BUG* threat model. Attacks are always balanced decisions, and if we leave ethics behind, anyone will mount an attack if profit outdoes risk. Therefore any agent must be allowed to switch from one role played in a session to another role in the next session.

It is perfectly conceivable in the new threat model that *B* decides to exploit his knowledge of *Na*. Because *A* believes to share that nonce only with *C*, agent *B* can later impersonate *C* with *A*, and retaliate against Lowe's authentication attack. For example, if *A* were also a bank, *B* can successfully ask a money transfer from *C*'s account to *B*'s by exhibiting *Na* and *Nb*. The bank *A* would believe the request came from *C*. This is a form of *retaliation attack*, which informally is an attack that another attack made possible. More precisely, it is a form of *indirect retaliation*: first, the bad *C* steals from the good *A* at the ugly *B*; in consequence, the bad *B* steals from the good *C* at the ugly *A*.

Retaliation may have important consequences both in terms of formal verification and in terms of protocol deployment. Verification has always pointed at establishing the existence of an attack, a point when the protocol was sent back to redesign. With retaliation in mind, the analysis must be continued even after an attack is found to realise whether it can be retaliated. This requires dealing with more than one quantifier on the same history and, most importantly, upgrading the underlying threat model into *BUG*. As I have experience in tool-supported protocol analysis, I have conducted some experiments in this vein, but I have to state that at present the mechanical tool support is not yet ready for analyses under *BUG*.

Protocol deployment may benefit from retaliation analyses. History has been seeing immense costs to reengineer and redeploy a protocol in order to fix it after an attack had been found. The social constraints deriving from retaliation may be of help. Who would steal money today under significant risk that this would cause twice as much stolen back to him tomorrow? It becomes interesting to precisely asses whether we can keep in operation those flawed protocols that admit retaliation. Attacking and immediately leaving the network is not profitable for the attacker in the long run and, most importantly, it can be traced.

Non-repudiation protocols must be mentioned now. They are meant to resist both the classical risk of an attacker's eavesdropping on two peers' channel, and the extra risk of the peers' illegally repudiating their own commitment to a transaction. At the extreme, one peer may want to deny payment after receiving the goods, and vice versa the other peer may want to deny the goods after receiving payment. But non-repudiation protocols deliver each peer valid evidence of the other's commitment [11]. It is clear that non-repudiation protocols must be analysed under *BUG* rather than under Dolev-Yao's threat model.

3.2 Formal Guarantees

I believe to have provided sufficient evidence that clear specifications of goals and threats are fundamental prerequisites to formal protocol analysis. However, it still remains to be assessed whether the very statements of the guarantees

that an analysis provides are meaningful, realistic or eventually of any use. It is not obviously true in practice that a conditional guarantee attains protocol correctness. Agents need to be able to evaluate in practice the truth value of the preconditions to be able to infer whether the postconditions hold. From the theoretical standpoint, it is certainly interesting to hear that if $P = NP$ then confidentiality is violated. But a protocol peer cannot evaluate that precondition. By contrast, he precisely needs to know when and on which intelligible evidence his session key can be considered confidential during the protocol run or thereafter. This is not a subjective view but a precise path to extra protocol insights, as I will show.

A real-world example can be easily derived from an e-commerce setting where a peer may want to repudiate his commitment to a transaction. Suppose that an electronic merchant vows in front of a notary public that he will ship the goods if he receives the relative 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 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.

These observations provided the basis for my principle of *goal availability*. It is a principle of realistic protocol analysis, that is, it aims at increasing the impact that a formal analysis can have on the real world. The principle is easy to describe informally, less easy to define precisely — see an early draft [10] or a chapter in my book [11]. Because a formal guarantee is beneficial to an agent only if the agent can verify its preconditions, goal availability exactly prescribes that the human analyser look for such kind of formal guarantees. I have always adopted this principle in my ten years of research in protocol verification even if I have only spelled out its precise definition lately. It seems fair to state that its very application has provided a number of novel protocol insights. And it can also be argued that goal availability has been somewhere up in the air to implicitly inspire other researchers.

My first intuition on goal availability dates back to my work on Kerberos IV in 1998 [15]. Kerberos comprises two trusted servers: the first issues session keys, say, of type one, the second issues session keys, say, of type two. This is a drastic simplification indeed. A key of type one is used to encrypt various keys of type two. I proved a guarantee that, if an agent received a key of type two, he could consider it confidential with the extra assumption that not too long before had the first server issued the key of type one used to encrypt the mentioned key of type two. Notice that “not too long before” had a precise meaning. My colleagues and I were content with this guarantee for some time. But soon I came to realise that its relevance was somewhat theoretical because never in practice would the agent be able to verify all its preconditions. No legal agent can inspect, within a realistic threat model, what happens at other network ends.

It has always been the case that, when a guarantee *cannot be made available to a peer*, namely it is impossible to have it conformed to the principle of goal availability, the protocol hides a flaw. It means that whenever an agent cannot

base its beliefs on clear facts whose truth value he can establish without ambiguity, he typically becomes forced to unsupported decisions that eventually the attacker exploits as targets. Kerberos IV made no exception: it required a correction to the first server’s operation. In this paper I would like to detail a subsequent outcome [11], on the Otway-Rees protocol [36]. It is a simple session key distribution protocol based on shared-key encryption.

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

An initiator A begins by contacting a responder B with a ciphertext built with its long-term key. It contains the session peers, a fresh nonce of the initiator’s, and a session identifier M . Message 2 sees B forward to a trusted server A ’s ciphertext along with his own, built the same way. The server sends B two tickets containing the same key in message 3. One gives the session key to B , while the other one is meant to give it to A , and B forwards it on in message 4.

If we focus on the second round (the last two messages), it appears that the protocol achieves its goal of key distribution: B learns the session key from receiving message 3, and A from message 4. With goal availability in mind, I was interested in verifying whether this goal can be proved on assumptions that the agents can verify. It was encouraging to find out that some work in the BAN logic was already available at the time: “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” [19]. Its second clause provides a negative answer to my concern. It means that the protocol fails to make key distribution available to either of its peers, but I meant to double-check this by myself. In my experience, that failure had to hide a flaw, but also this intuition had to be verified.

I hence began to experiment with the formal analysis of this protocol using the Inductive Method, which I outline in the next section. My experiments are published with the 2006 distribution of Isabelle [39]. It was unsurprising to find out that it is impossible to prove a conjecture useful to B : if B receives message 3 and sends the corresponding message 4 to A , then A knows the session key. Any proof attempt hangs in front of a simple flaw that sees the attacker prevent delivery of message 4 (so A never learns the key). No weaker conjectures, based on assumptions that B can check, confirming that A knows the session key are provable, so my conclusion is that the protocol indeed fails to make key distribution available to B . And the flaw I just mentioned is the price to be paid. Notice, however, that without the guidance of goal availability, one would be tempted to make the extra assumption that A received message 4 and conclude the proof. That conjecture would become a theorem confirming a guarantee that is not useful to B because he cannot assess all its assumptions in practice.

My conclusion about A was similar but had much more exciting consequences. In brief, the protocol fails to make key distribution available to A

because, whenever A receives the session key in message 4, it might be the case that the attacker prevented delivery of the corresponding message 3 to B (so B never learns the key). The attacker would merely have to extract the ticket for A from message 3 and forward it to A with the session identifier M . Also in this case, however, ignoring goal availability might induce a researcher to make the extra assumption that B receives message 3. This would produce a theorem that cannot and must not be evaluated as a useful guarantee for A . I was happy that my experiments confirmed the second clause of the BAN statement.

Similar mistakes can be found in the literature. The BAN paper reports a theorem about the shared-key Needham-Schroeder protocol meant to signify authentication of A with B . Its proof had to make an extra assumption to proceed, that a message received by B were fresh, namely not an attacker’s replay. This was correctly reported as a “dubious” practice — in fact, it is perfectly against goal availability because B cannot ever verify that such an assumption is met. The guarantee was generally accepted but a flaw was soon published describing how the attacker can impersonate A with B in a realistic threat model [23]. The flaw was exactly due to B ’s chance of erroneously evaluate the extra assumption.

Going back to the BAN statement on the Otway-Rees protocol, I went on to parse its first clause: it could easily be confirmed by inspecting the protocol. But being fond of linguistics, I got particularly interested in verifying the causal relationship introduced by the conjunction “so” (the International Dictionary of English by the Cambridge University Press explains “so” as “and for that reason”). That relationship seems to imply that the protocol cannot inform any of its peers that the other one knows the session key *because* none of them ever uses the key, once he has received it, to encrypt anything. What’s more is that this seemed to be formulated as a general lesson about the class of protocols that do not prescribe to use the key once it is received. I soon rejected this lesson as my previous work on Kerberos already supported that protocol as a counterexample. But it seemed challenging to experiment directly with Otway-Rees.

I realised that B could not be helped. Because he is the peer who forwards the ticket (containing the session key) to the other one, he cannot be informed that A knows it unless A uses it to encrypt something intelligible to send over for B to check. In this extent, I found the causality in the BAN statement correct. Because A ’s situation seems more advantageous, it seemed interesting to verify the outcome of the following modification to message 3.

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

The only discrepancy with the original message is a single parenthesis shift: encryption with B ’s key now is the outermost operator. Notice that also my variant protocol “does not make use of Kab as an encryption key”. However, I could prove the following theorem of its inductive model: if A learns the session key from receiving message 4, then B knows that session key. This theorem confirms that the variant protocol makes the goal of key distribution available to A , although no peer uses the key to encrypt anything.

My findings refute the BAN statement — precisely, its half concerning A . It teaches that in general the formal guarantees about security protocols must be designed and interpreted with extreme care. And in particular, it shows that not only does working with goal availability prevent erroneous interpretations of the formal guarantees, but also helps unveil unknown protocol niceties. Other examples of the usefulness of goal availability exist [11].

4 A Formal Method of Protocol Verification

I have built the vast majority of my experience in protocol verification using the Inductive Method, which is mechanised in the proof assistant Isabelle [35]. This section begins (§4.1) with a brief outline of the method (see [11, 37] for a complete presentation), and terminates (§4.2) with a demonstration of some of the findings described above.

4.1 The Inductive Method

Figure 1 shows file `fragment.thy` opened by the Isar graphical interface to Isabelle [40]. The file sums up the definitions of a few main functions.

There exists an unlimited population of principals who are entitled to initiate at will an unlimited number of sessions of the given security protocol. Among the principals is the *spy*, who monitors the entire network traffic and in consequence knows who sends and who receives which messages. This feature indicates that the method was conceived under Dolev-Yao’s threat model. An unspecified set of *bad* principals have colluded with the spy by revealing their long-term secrets. The spy is herself bad, as it can be seen in Figure 1. However, she is the only network principal who can send arbitrary messages built from components intercepted from the network traffic. Interception is modelled by the function `knows`, and creation of fake messages by a conjunct use of the functions `analz` and `synth`. All are described below.

The network traffic develops according to the events performed by the principals while they are executing the given protocol. Typical events are to send or to receive a message. 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 `Says B A m'` each time the trace contains the event `Gets B m`. In other words, that reception event is a precondition and that sending event is a postcondition of the rule. Therefore, the events occur via the firing of the inductive rules. But, as induction prescribes, no rule is forced to fire, so no event is forced to occur.

The binary function `knows`, defined by primitive recursion in Figure 1, formalises the knowledge that principals derive from observing a trace [11]. So, `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. Figure 1 also shows that 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. This means that it is assumed that no cryptanalysis is possible, namely that encryption is totally reliable. In consequence, confidentiality of a message component *m* in a trace *evs* can be expressed as

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

```

emacs: fragment.thy
File Edit View Cmds Tools Options Buffers Proof-General Isabelle X-Symbol Help
State Context Retract Undo Next Use Goto Find Command Stop Restart Info Help

fragment.thy
consts
  bad :: "agent set"
specification (bad)
  Spy_in_bad [iff]: "Spy ∈ bad"

consts parts :: "msg set ⇒ msg set"
inductive "parts H"
  intros
    Inj [intro]: "X ∈ H ⇒ X ∈ parts H"
    Fst: "⟨X, Y⟩ ∈ parts H ⇒ X ∈ parts H"
    Snd: "⟨X, Y⟩ ∈ parts H ⇒ Y ∈ parts H"

consts analz :: "msg set ⇒ msg set"
inductive "analz H"
  intros
    Inj [intro, simp]: "X ∈ H ⇒ X ∈ analz H"
    Fst: "⟨X, Y⟩ ∈ analz H ⇒ X ∈ analz H"
    Snd: "⟨X, Y⟩ ∈ analz H ⇒ Y ∈ analz H"
    Decrypt [dest]: "[[Crypt K X ∈ analz H; Key(invKey K) ∈ analz H]
      ⇒ X ∈ analz H]"

consts synth :: "msg set ⇒ msg set"
inductive "synth H"
  intros
    Inj [intro]: "X ∈ H ⇒ X ∈ synth H"
    Agent [intro]: "Agent agt ∈ synth H"
    Number [intro]: "Number n ∈ synth H"
    Hash [intro]: "X ∈ synth H ⇒ Hash X ∈ synth H"
    MPair [intro]: "[X ∈ synth H; Y ∈ synth H] ⇒ ⟨X, Y⟩ ∈ synth H"
    Crypt [intro]: "[X ∈ synth H; Key(K) ∈ H] ⇒ Crypt K X ∈ synth H"

consts
  knows :: "agent ⇒ event list ⇒ msg set"
primrec
  knows_Mil: "knows A [] = initState A"
  knows_Cons:
    "knows A (ev # evs) =
      (if A = Spy then
        (case ev of
          Says A' B X ⇒ insert X (knows Spy evs)
        | Gets A' X ⇒ knows Spy evs
        | Notes A' X ⇒
          if A' ∈ bad then insert X (knows Spy evs) else knows Spy evs)
        else
        (case ev of
          Says A' B X ⇒
            if A'=A then insert X (knows A evs) else knows A evs
        | Gets A' X ⇒
            if A'=A then insert X (knows A evs) else knows A evs
        | Notes A' X ⇒
            if A'=A then insert X (knows A evs) else knows A evs))"

IS08-----XEmacs: fragment.thy (Isar script XS:isabelle/s Font)-----All-----

```

Fig. 1. Defining the main functions for the Inductive Method in Isabelle

The function `synth`, also defined by induction in Figure 1, is crucial. It expresses the spy’s illegal activity in building up messages at will. It can be seen that the spy can synthesise any agent name or number (timestamp), and hash available messages. She can also concatenate messages into longer ones, and build ciphertexts using available keys. Therefore, the set

$$\text{synth}(\text{analz}(\text{knows Spy } \textit{evs}))$$

expresses all messages that the spy can synthesise from the analysis of the network traffic over trace *evs*.

Each kind of cryptographic key has its own syntax: the long-term keys shared with the trusted server are denoted by function `shrK`. Moving on to the actual formal guarantees, they come in the form of theorems that hold of the protocol model. Precisely, each theorem is expressed over a generic trace and hence holds in general. A proof is conducted by structural induction on the length of the trace, resulting in a number of long subgoals that can span several pages. Isabelle solves the simple cases automatically.

4.2 Verifying the Otway-Rees-Bella Protocol

I used the Inductive Method to analyse my variant of the Otway-Rees protocol mentioned in the previous section. Its inductive model, which is available with the 2006 distribution of Isabelle [39], can be read in Figure 2.

It can be seen that the full specification comes with file `OtwayReesBella.thy` and is defined as the Isabelle theory `OtwayReesBella`. This is built up by extending theory `Public`, which defines all functions for long-term cryptographic keys. The actual protocol model, constant `orb`, is declared as a set of traces and defined inductively by five rules. Rule `Nil` sets the base of the induction stating that the empty trace belongs to the model. Rule `Fake` models the spy’s activity: given a trace *evsf* in the model, its extension (`#` is the list append operator) with the event whereby the spy sends some agent *B* one of her fake messages *X* is still a trace of the protocol model. The inductive layout is clear. Notice that the message *X* is derived from the set of fakes described in the previous section. The remaining rules model the steps of the protocol, one by one.

Rule `ORB1` has the only premise that *A* uses a fresh nonce. Freshness on a trace is modelled via the function `used`, whose intuitive definition I omit from this paper. Rule `ORB2` also relies on the precondition that *B* received an instance of message 1 on the given trace *evs2*. Its postcondition is that *B* sends message 2 to the server. Rule `ORB3` describes the server’s operation upon reception of a well-formed request. It issues a fresh session key and sends message 3 to *B*. Rule `ORB4` sees *B* complete the protocol by sending the last message to *A*. This is subjected to three preconditions: that *B* invoked the server, that he received a matching reply, and that the message component *X* is a ciphertext. It is the ticket that the server issued for *A*, and that *B* cannot decrypt. In practice, he identifies it from its length, while the model insists that *X* cannot be written as any concatenated message. I omit two rules for simplicity: one states that a

```

emacs: OtwayReesBella.thy
File Edit View Cmds Tools Options Buffers Proof-General Isabelle X-Symbol Help
State Context Retract Undo Next Use Goto Find Command Stop Restart Info Help
OtwayReesBella.thy
ID:      $Id: OtwayReesBella.thy,v 1.1 2006/02/01 14:22:15 paulson Exp $
Author:  Giampaolo Bella, Catania University *)

header(*Bella's version of the Otway-Rees protocol*)

theory OtwayReesBella imports Public begin

consts orb :: "event list set"
inductive "orb" intros

Nil: "[ ] ∈ orb"

Fake: "[[evsf ∈ orb; X ∈ synth (analz (knows Spy evsf))]]
      ⇒ Says Spy B X # evsf ∈ orb"

ORB1: "[[evs1 ∈ orb; Nonce NA ≠ used evs1]]
      ⇒ Says A B {Nonce M, Agent A, Agent B,
                  Crypt (shrK A) {Nonce NA, Nonce M, Agent A, Agent B}}
      # evs1 ∈ orb"

ORB2: "[[evs2 ∈ orb; Nonce NB ≠ used evs2;
          Gets B {Nonce M, Agent A, Agent B, X} ∈ set evs2]]
      ⇒ Says B Server {Nonce M, Agent A, Agent B, X,
                       Crypt (shrK B) {Nonce NB, Nonce M, Nonce M, Agent A, Agent B}}
      # evs2 ∈ orb"

ORB3: "[[evs3 ∈ orb; Key KAB ≠ used evs3;
          Gets Server
            {Nonce M, Agent A, Agent B,
             Crypt (shrK A) {Nonce NA, Nonce M, Agent A, Agent B},
             Crypt (shrK B) {Nonce NB, Nonce M, Nonce M, Agent A, Agent B}}
          ∈ set evs3]]
      ⇒ Says Server B {Nonce M,
                       Crypt (shrK A) {Crypt (shrK A) {Nonce NA, Key KAB}, Nonce NB, Key KAB}}
      # evs3 ∈ orb"

(*B can only check that the message he is bouncing is a ciphertext*)
ORB4: "[[evs4 ∈ orb; B ≠ Server; ∀ p q. X ≠ {p, q};
          Says B Server {Nonce M, Agent A, Agent B, X},
          Crypt (shrK B) {Nonce NB, Nonce M, Nonce M, Agent A, Agent B}}
          ∈ set evs4;
          Gets B {Nonce M, Crypt (shrK B) {X, Nonce NB, Key KAB}} ∈ set evs4]]
      ⇒ Says B A {Nonce M, X} # evs4 ∈ orb"

IS08-----XEmacs: OtwayReesBella.thy (Isar script XS-isabelle/s Font)-----Top-----

```

Fig. 2. Inductive model of the Otway-Rees-Bella protocol

message that is sent can be received, and the other one allows session keys to be accidentally lost to the spy.

A number of theorems proved of the original protocol model continue to hold about my variant. The main theorem that holds only of my variant can be found in Figure 3. Its statement is simple: if A initiated the variant protocol with B and received an instance of the last message that delivers a session key to her, then her peer B knows the same session key. As A can verify the preconditions, this theorem provides a guarantee that the variant protocol makes key distribution available to A .

The proof is simple to develop in a forward style. The first command uses the second assumption to derive that the ticket can be analysed from the traffic,

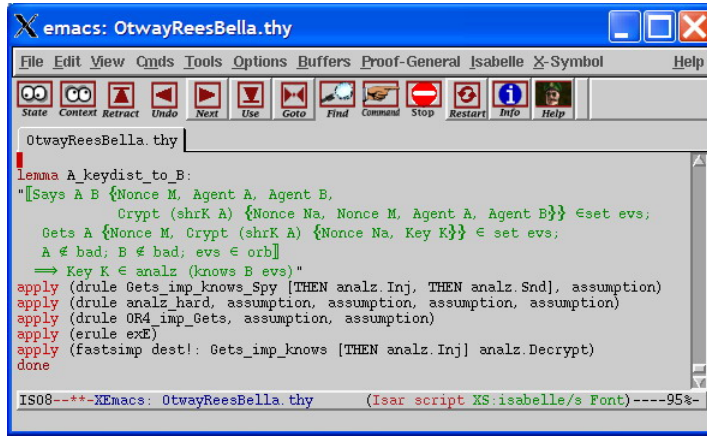


Fig. 3. Proving key distribution for A in the Otway-Rees-Bella protocol

namely that

$$\text{Crypt}(\text{shrK } A)\{\text{Nonce } Na, \text{Key } K\} \in \text{analz}(\text{knows Spy } evs).$$

This fact is stronger than the analogous one (which also holds) expressed in terms of `parts` rather than of `analz`, because in general $\text{analz } H \subseteq \text{parts } H$. In intuitive terms, this stronger fact implies B 's activity of extracting A 's ticket from message 3, thereby learning the session key from her own ticket, and of sending message 4. By contrast, the weaker fact in terms of `parts` is insufficient, as it also holds before B receives message 3. Notice that this argument fails for the original protocol. In technical terms, what concludes that B sent message 4 on evs is the application of lemma `analz_hard` by the second command. The third command applies another lemma saying that, if B sent message 4, he received message 3. The rest of the proof easily deduces that B learnt the session key, which is the thesis.

Is the proof that simple? As it is often the case with theorem proving, much proof efforts are deferred to the necessary lemmas. My example is emblematic, as it only requires a bunch of commands to conduct a short forward-style proof. But the subsidiary lemma `analz_hard` requires an inductive proof whose current distribution terminates at level 26! Its development required about three quarters of the total time I spent on the protocol, which was five (man) weeks. The proof details certainly lie outside the focus of this paper.

It was heartening to realise that goal availability is reasonably easy to use within the Inductive Method. I have specified elsewhere [11] that this is not trivially the case with any formal method. Both the protocol specification and the reasoning about it must be conducted from each agent's viewpoint.

5 Open Issues

There exist a variety of open issues currently related to the problem of security protocol correctness — which I wish I had time to investigate all! I outline just a few of them here, and make an anchor citation each time.

Multi-protocol attacks. Each security protocol is normally analysed, whether formally or informally, as a stand-alone object. It is implicitly assumed that the protocol is always run in isolation, without any interaction with other protocols. However, a protocol that is secure if run in isolation may no longer be so if it can be interleaved with other protocols, as it is realistic at present. Bad agents may attempt to exploit the messages of a protocol to violate another protocol [21]. This issue deserves attention both at time of design and especially in terms of formal verification.

Secure protocol composition. There exists an even more general issue, which is composition of correct security protocols. System composition is not necessarily property-preserving. But with protocols getting more and more complex, it is getting unfeasible to analyse each protocol as a monolithic object down to each component. The sheer size of the system is often the major obstacle. Some efforts exist using the probabilistic approach [29]. When I studied a protocol for certified e-mail that adopts SSL [16], I made the assumption that no low-level interaction between the two protocols was possible. If this were unsound, reusability of existing proofs would be unfeasible, a severe constraint to the entire area.

Reconciling the two approaches to protocol verification. The probabilistic and black-box cryptography approaches still partition the international community of researchers. I already mentioned two important contributions by Abadi [2] and by Gollmann [26], and it may seem emblematic that they are not very recent. However, last year saw an important contribution by Backes and Laud [7], and this year's web site of an important security conference features what seems a relevant programme title [18]. Its proceedings are unavailable yet, but I am seriously hoping that we are facing a renewed interest. To be fair, I recall some visitor of the Computer Laboratory at Cambridge University attempting to embed exclusive-or encryption in the Inductive Method, though in vain. I set my own goals in another direction [11].

Probabilistic approach for e-commerce protocols. I believe to have contributed to what qualifies as the formal analysis of the most entangled security protocol tackled thus far. It is the purchase phase of SET [14]. It uses all available cryptographic measures, such as symmetric and asymmetric cryptography, public-key infrastructures, cryptographic hashing, digital envelopes and dual signatures. Its goals are complex distributed agreement properties between more than two peers, as also other researchers noted [34]. It is worth investigating the contributions that the probabilistic approach may give to the analysis of complex e-commerce protocols like SET and of their subtle goals.

Security of non-hierarchical networks. Classical wired networks are hierarchical: each node in a LAN sees the Internet via its boundary router, which is in turn connected to a more central node serving a number of LANs, and so on. The formation of an IP address is emblematic. Conversely, peer-to-peer networks bear a flat structure by nature, as they are formed by a number of nodes wishing to enjoy (and provide) the same service. In particular, mobile ad-hoc networks (known as MANETs) seem the very opposite of classical wired networks. Security in this context is a multifaceted concern [28]. It can no longer be assumed that a trusted third party (TTP) or a certification authority (CA) or a key-distribution centre (KDC) are available, because nodes can join or leave frequently. The management of trust relationships among participating nodes is most important, with some protocols currently under scrutiny [4]. Also the protocols aimed at secure communication, such as WEP or WPA, need analysis and improvements.

Complete formal treatment of retaliation. A complete formal treatment of retaliation [13] is still out of reach. As a start, it demands disposing with Dolev-Yao's model and adopting BUG. Then, it requires a detailed formalisation of a retaliation attack: if there exists a trace with a classical attack, it must be studied the existence of a trace that continues the first one and features a retaliation attack. The impossibility of retaliation requires inspecting all possible continuations of the first trace, producing alternation of quantifiers. It may become particularly elucidating to implement these concepts on model checkers, tools that have importantly contributed to the area thus far [31].

Machine support for non-crisp analyses. Real-world concepts are often non-crisp, meaning that they are not achieved in their entirety but, rather, are attained up to a certain level. For example, each resource can be labelled to express its sensitivity in terms of confidentiality, such as “top-secret”, “secret”, “restricted”, “public”. Exhibiting a highly confidential item also provides a high level of authentication of its holder, whereas in practice we still tend to attach some level of authentication to an e-mail received via HTTP. I used the framework of soft-constraint programming to formally analyse static configurations of security protocols in their non-crisp goals [12]. However, that work requires machine assistance, even if a different formal approach were necessary.

Anti-DoS techniques. Denial of service (DoS) is a subtle threat to our time. A multitude of legitimate requests can overflow the resources of a server in terms of memory, computation or bandwidth. I find a denial-of-service attack of peculiar nature: it is illegal merely because it is an indefinite replication of legal requests. The technique of *cookie-transformation* [22] attempts to delay the computational burden of a server in a client-to-server session to when it is clear that also the client is computationally active at an address. It is adopted in common protocols such as IKE and JFK, which therefore require an initial hello round between server and client. It is debated whether these and similar strategies qualify as viable and general solutions. It is still unclear whether and how formal verification can contribute to address this matter.

6 Conclusions

A security protocol is correct if it lives up to the goals that its designers stated against specific threats. I have demonstrated the importance of defining both goals and threats in detail.

I have also described the importance of a threat model (called BUG) that seems more appropriate to our days. For example, it is necessary to analyse non-repudiation protocols.

The new threat model allows the concept of retaliation attack. If attackers' logic merely is profit, they may decide to refrain from attacking if their actions can be retaliated.

The formal analysis of security protocols can address various issues. However, it must be conducted with care. Adopting the principle of goal availability can favour extra insights.

The Inductive Method effectively helps to analyse security protocols under the classical threat model and with the mechanical support of the proof assistant Isabelle, which comes under the BSD licence.

A variety of issues concerning protocol correctness remain open at present. They range from the need to mechanically support the novel concepts to the conjugation of different approaches to protocol analysis.

Research in protocol correctness has been attracting wide interest from a number of areas in Informatics for nearly three decades by now. The international ferment does not seem to settle.

Acknowledgments

I owe a great lot to the interaction that colleagues have variously offered me throughout the years: research groups, projects, program committees, even anonymous reviewers, and especially coauthors.

References

1. M. Abadi and B. Blanchet. Computer-assisted verification of a protocol for certified email. In R. Cousot, editor, *Proc. of the 10th International Symposium on Static Analysis (SAS'03)*, LNCS 2694, pages 316–335. Springer, 2003.
2. M. Abadi and P. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). In *Proc. of the International Conference IFIP on Theoretical Computer Science (TCS'00)*, pages 3–22. Springer, 2000.
3. M. Abdalla, P. A. Fouque, and D. Pointcheval. Password-based authenticated key exchange in the three-party setting. *IEE Proceedings Information Security*, 153(1):27–39, 2006.
4. K. Aberer and Z. Despotovic. Managing trust in a peer-2-peer information system. In *Proc. of the 10th International Conference on Information and Knowledge Management (CIKM'01)*, pages 310–317. ACM Press, 2001.
5. R. Anderson and R. M. Needham. Programming Satan's computer. In J. Van Leeuwen, editor, *Computer Science Today: Recent Trends and Developments*, LNCS 1000, pages 426–441. Springer, 1995.

6. A. Armando and L. Compagna. An optimized intruder model for SAT-based model-checking of security protocols. In *Proc. of the Workshop on Automated Reasoning for Security Protocol Analysis (ARSPA'04)*, ENTCS 125, pages 91–108. Elsevier Science, 2005.
7. M. Backes and P. Laud. Computationally sound secrecy proofs by mechanized flow analysis. In *Proc. of the 13th ACM conference on Computer and communications security (CCS'06)*, pages 370–379. ACM Press, 2006.
8. D. Basin, S. Mödersheim, and L. Viganò. An on-the-fly model-checker for security protocol analysis. In E. Sneekenes and D. Gollmann, editors, *Proc. of the 8th European Symposium on Research in Computer Security (ESORICS'03)*, LNCS 2808, pages 253–270. Springer, 2003.
9. D. Basin, S. Mödersheim, and L. Viganò. OFMC: A symbolic model-checker for security protocols. *International Journal of Information Security*, 4(3):181–208, 2005.
10. G. Bella. Availability of protocol goals. In B. Panda, editor, *Proc. of the 18th ACM Symposium on Applied Computing (ACM SAC'03)*, pages 312–317. ACM Press, 2003.
11. G. Bella. *Formal Correctness of Security Protocols*. Information Security and Cryptography. Springer, 2007.
12. G. Bella and S. Bistarelli. Soft constraint programming to analysing security protocols. *Theory and Practice of Logic Programming*, 4(5):1–28, 2004.
13. G. Bella, S. Bistarelli, and F. Massacci. Retaliation: Can we live with flaws? In M. Essaidi and J. Thomas, editors, *Proc. of the Nato Advanced Research Workshop on Information Security Assurance and Security*. IOS Press, 2005.
14. G. Bella, F. Massacci, and L. C. Paulson. Verifying the SET purchase protocols. *Journal of Automated Reasoning*, 36(1-2):5–37, 2006.
15. G. Bella and L. C. Paulson. Kerberos Version IV: Inductive analysis of the secrecy goals. In J.-J. Quisquater, Y. Deswarte, C. Meadows, and D. Gollmann, editors, *Proc. of the 5th European Symposium on Research in Computer Security (ESORICS'98)*, LNCS 1485, pages 361–375. Springer, 1998.
16. G. Bella and L. C. Paulson. Accountability protocols: Formalized and verified. *ACM Transactions on Information and System Security*, 9(2):1–24, 2006.
17. M. Bellare and P. Rogaway. Provably secure session key distribution — the three party case. In *Proc. of the 27th ACM SIGACT Symposium on Theory of Computing (STOC'95)*, pages 57–66. ACM Press, 1995.
18. B. Blanchet. Computationally sound mechanized proofs of correspondence assertions. <http://www.dsi.unive.it/CSF20/program.csf20.html>.
19. M. Burrows, M. Abadi, and R. M. Needham. A logic of authentication. *Proc. of the Royal Society of London*, 426:233–271, 1989.
20. E. Cohen. First-order verification of cryptographic protocols. *Journal of Computer Security*, 11(2):186–216, 2003.
21. C. J. F. Cremers. Feasibility of multi-protocol attacks. In *Proc. of the First International Conference on Availability, Reliability and Security (ARES'06)*, pages 287–294. IEEE Press, 2006.
22. A. Datta, A. Derek, J. Mitchell, and D. Pavlovic. A derivation system for security protocols and its logical formalization. In *Proc. of the 16th IEEE Computer Security Foundations Workshop (CSFW'03)*, pages 109–125. IEEE Press, 2003.
23. D. E. Denning and G. M. Sacco. Timestamps in key distribution protocols. *Communications of the ACM*, 24(8):533–536, 1981.
24. D. Dolev and A. Yao. On the security of public-key protocols. *IEEE Transactions on Information Theory*, 2(29):198–208, 1983.

25. D. Gollmann. What do we mean by entity authentication? In *Proc. of the 15th IEEE Symposium on Security and Privacy (SSP'96)*, pages 46–54. IEEE Press, 1996.
26. D. Gollmann. On the verification of cryptographic protocols — a tale of two committees. In *Proc. of the Workshop on Secure Architectures and Information Flow, ENTCS 32*. Elsevier Science, 2000.
27. J. D. Guttman, F. J. Thayer, J. A. Carlson, J. C. Herzog, J. D. Ramsdell, and B. T. Sniffen. Trust management in strand spaces: A rely-guarantee method. In D. Schmidt, editor, *Proc. of the 13th European Symposium on Programming (ESOP'04)*, LNCS 2986, pages 325–339. Springer, 2004.
28. T. Jiang and J. S. Baras. Ant-based adaptive trust evidence distribution in manet. In *Proc. of the 24th International Conference on Distributed Computing Systems (ICDCSW'04)*, pages 588–593. IEEE Press, 2004.
29. E. Kushilevitz, Y. Lindell, and T. Rabin. Information-theoretically secure protocols and security under composition. In *Proc. of the 38th annual ACM Symposium on Theory of Computing (STOC'06)*, pages 109–118. ACM Press, 2006.
30. G. Lowe. An attack on the Needham-Schroeder public-key authentication protocol. *Information Processing Letters*, 56(3):131–133, 1995.
31. G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using CSP and FDR. In T. Margaria and B. Steffen, editors, *Proc of the 2nd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'96)*, LNCS 1055, pages 147–166. Springer, 1996.
32. G. Lowe. Some new attacks upon security protocols. In *Proc. of the 9th IEEE Computer Security Foundations Workshop (CSFW'96)*. IEEE Press, 1996.
33. G. Lowe. A hierarchy of authentication specifications. In *Proc. of the 10th IEEE Computer Security Foundations Workshop (CSFW'97)*, pages 31–43. IEEE Press, 1997.
34. C. Meadows and P. Syverson. A formal specification of requirements for payment transactions in the SET protocol. In R. Hirschfeld, editor, *Proc. of Financial Cryptography '98*, LNCS 1465. Springer, 1998.
35. T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*. Springer, 2002. LNCS Tutorial 2283.
36. D. Otway and O. Rees. Efficient and timely mutual authentication. *ACM Operating Systems Reviews*, 21(1):8–10, January 1987.
37. L. C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6:85–128, 1998.
38. P. Syverson. A taxonomy of replay attacks. In *Proc. of the 7th IEEE Computer Security Foundations Workshop (CSFW'94)*, pages 187–191. IEEE Press, 1994.
39. URL. Isabelle download page.
<http://www.cl.cam.ac.uk/Research/HVG/Isabelle/download.html>.
40. URL. Proof General: a generic interface for proof assistants.
<http://proofgeneral.inf.ed.ac.uk>.