What is Protocol Analysis?

Francien Dechesne, Jan van Eijck, Wouter Teepe, Yanjing Wang

The following is a transcript of one of the discussion sessions that took place during the Workshop on **Games, Action and Social Software** *at the Lorentz Center in Leiden. The discussion theme was set by the workshop organizers: "Is logic useful for the analysis of protocols, and if so, how?" The theme has attracted the usual protagonists, plus a cognitive scientist and a specialist in computer security.*

Logician: The workshop organizers have asked me to chair this session. I suppose the first thing we should establish is that everyone present here understands the question at hand in the same way.

Philosopher: Well, I could definitely use some clarification. A *protocol* is generally understood to be some set of rules or conventions, but that is very broad. I'm afraid I do not have a clear picture of what is meant by *protocol analysis*. Could anyone explain this to me?

Cognitive Scientist: In cognitive science, protocol analysis is the name of an experimental method for gathering so-called intermediate state evidence concerning the procedures used by a human to compute a function. More in particular, subjects are trained to think aloud as they solve a problem, and their verbal behavior forms the basic data to be analyzed.

Philosopher: This sounds as if the analysis is understood as *reconstructing* or mimicking the protocol that is (assumed to be) used by this human. Do I see this correctly? Maybe you could give a more concrete example.

Cognitive Scientist: Well, standard examples of protocols we analyze are the mechanisms people use to solve brainteaser problems. You can think of the following "crypt-arithmetical problem": *(Writes on the whiteboard.)*

$$\begin{array}{cccccccc} D & O & N & A & L & D & & \\ G & E & R & A & L & D & + & \\ \hline R & O & B & E & R & T & & \end{array}$$

Let it be given that $D = 5$. How do you solve this?

Computer Scientist: May I? Let us see. If $D = 5$ then $T = 0$, from the last column. From the first column we know that $G + 5 = R$, so $R \geq 5$. From the second column we then get that E has to be 0. This gives $A = 5$ from column four, and there has to be a carry to column three. Looking at the fifth column, we see that $L = 2$ would give $R = 5$ and $G = 0$. Let's pursue this. In column three, we could set $N = 1$, which yields $B = 7$, as there was a carry from column four. The value of O is not constrained. Let's set it to 1. We are done:

$$\begin{array}{cccccccc} 5 & 1 & 1 & 5 & 2 & 5 & & \\ 0 & 0 & 5 & 5 & 2 & 5 & + & \\ \hline 5 & 1 & 7 & 0 & 5 & 0 & & \end{array}$$

But this solution is far from unique. As remarked, we could have picked any other value for O to get a solution. Also, we could have chosen $L = 3$, which would have given $R = 7$ and $G = 2$, again for different values of O. Or $L = 4 \ldots$

Philosopher: Yes, I think we get the point. In cognitive science, the next step of what you call protocol analysis would be to analyze this verbal description of how our computer scientist solved the puzzle.

Cognitive Scientist: Yes, we would now use this so-called *verbal protocol* to infer the subject's problem space.

Philosopher: What do you mean by "problem space"?

Cognitive Scientist: The problem space is a set of rules that are used to transform knowledge states concerning the problem.

Philosopher: So these are just postulated?

Cognitive Scientist: That's right. Once one has a suitable problem space, one can proceed to create a problem behavior graph, supposed to reflect state transitions as subjects search through the problem space in their attempts to solve the problem.

Computer Scientist: I see what you mean. So a problem space is just a labeled transition system viewed as a search space. If you give me a finite search space I can give you any number of solution algorithms for your problem.

Cognitive Scientist: Yes, and the task of protocol analysis is to pinpoint the particular search algorithms that reflect how humans do it. Finally, one can compare the verbal protocol to the computer simulation, to validate the assumptions that led to the simulating program. Once validated, the simulating program provides a rich description of the processing steps performed by the human subject. Two classic books dealing with the subject are [4] and [7].

Logician: OK, I think this interpretation of term "protocol analysis" is clear now. But I have the feeling the term has quite a different meaning in computer science, is that right?

Security Analyst: Indeed, protocol analysis in computer science refers to a quite different activity. In computer security, we are concerned with the design and specification of communication protocols, and with checking whether they fulfill the given goal and/or satisfy the desired properties.

Any protocol that is designed to satisfy some kind of *security* property, can be called a security protocol. Such properties may be about trust, or about fairness (making sure that agents can't cheat), but for this discussion, maybe it's best to restrict ourselves a bit. A more narrow class of security protocols would be the *communication* protocols (i.e. sets of rules for sending and receiving messages), in which it is ensured that particular pieces of information are kept *secret* from certain parties, or in which it is ensured that you're talking to the agent you intend to talk to: *authentication*.

Logician: So security protocols are formal constraints on communication patterns that are meant to ensure some agents get to know something while outsiders remain in the dark, or that ensure that you know whom you're talking to. It sounds as if the protocols ensure certain epistemic properties of the communication. This relates to our central question, whether logic can be useful for protocol analysis. It seems we have a perfect working place here for epistemic logic: the logic for reasoning about knowledge.

Cognitive Scientist: I am sorry, but I don't really understand. I have trouble with vague formulations like "get to know something" and "outsiders". What is it that should be analyzed? The protocol? Properties of the protocol, properties of the parties participating in the protocol?

Philosopher: Actually, there was a talk at this workshop addressing such questions, by Francien Dechesne and Yanjing Wang. It was about how security protocols can be analyzed, and a link was made with dynamic epistemic logic. As an example of a security protocol, they mentioned the Needham-Schroeder public key authentication protocol. Maybe it's good to use that example to clarify the terminology a bit?

Cognitive Scientist: That's a good idea, for I missed that talk, I am afraid. What is this Needham-Schroeder protocol supposed to do? And how does it achieve its purpose?

Security Analyst: Before we state the actual protocol, we need some preliminary notions and assumptions. For one thing, we assume that every agent a owns a private (secret) key to decrypt messages that were encrypted with the publicly available key PK_a. By the way, does everyone know how public key encryption works?

Computer Scientist: Shall I explain? Public key encryption is a nice example of how work in pure mathematics (in this case, number theory) may suddenly and unexpectedly turn out to have high practical relevance. Public key encryption is based on the existence of mathematical operations that are very difficult to reverse. For example, multiplication of two large prime numbers is easy, but finding the prime factors of a very large number is extremely difficult: if I tell you, for instance, that 7879 and 5113 are primes, then it is very easy for you to calculate their product. But suppose instead of this I tell you that 40285327 is the product of two primes, and challenge you to produce these primes...

Cognitive Scientist: But if you give me one factor, it is easy to find the other. So I guess the key is supposed to be some number or maybe other piece of information that makes the reversal operation very easy to do?

Computer Scientist: You got it. And it should be understood that for really large numbers, without the key, sophisticated guesses or even supercomputers would be of no help. No known method for finding the prime factors of a number is substantially better than trial and error.

Security Analyst: Yes, then with some tricks, for example by applying the RSA algorithm [8] we could have one-to-one functions f on the natural numbers with the property that computing $f(N)$ is easy if you know the "public key", even for very large N, while computing $f^{-1}(M)$ for large M is extremely

difficult without knowing the "secret key".

Actually, for our analysis, we don't have to go into all the mathematical details. We can leave that to the specialized mathematicians, the cryptographers. In security *protocol* analysis it is the custom to keep the mathematics behind the encryption outside the model. This is called the *black box approach* to cryptography. It is enough to just assume that such keys exist.

Logician: I see. So, the relevant part is that we may assume the existence of some "practically unbreakable" encryption functions, that allow anyone to encrypt a message—using the public key of the intended receiver—but only allows the possessor of the secret key to decrypt it.

Security Analyst: Right. Let me now write down the Needham-Schroeder protocol. Its purpose is to make sure that two agents who are communicating with each other can identify their correspondent. It assumes that the private keys are kept secret, and the public keys are available for all, so that every agent can create messages that can be read by one and only one agent. Here it is: *(Writes on the whiteboard:)*

$$\begin{array}{lll} \text{Message 1} & a \to b: & \{n_a, a\}_{\mathrm{PK}_b} \\ \text{Message 2} & b \to a: & \{n_a, n_b\}_{\mathrm{PK}_a} \\ \text{Message 3} & a \to b: & \{n_b\}_{\mathrm{PK}_b} \end{array}$$

Here n_a is a so-called *nonce*...

Cognitive Scientist: Nonce? Sounds like nonsense...

Security Analyst: In a sense it is *non-sense*. A nonce is a very big arbitrary number that a has privately generated. It is assumed that it is impossible for others to have a clever guess. How such numbers can be generated is also left to the cryptographers.

Cognitive Scientist: But still there's one thing rather unclear to me. Why doesn't the first message simply consist of a public-key encoded $\{a\}_{PK_b}$? Why the nonce? I mean, it's now encrypted, right?

Security Analyst: Well, this nonce serves as a kind of time stamp. It was generated just for the purpose of this particular message, as a kind of challenge for b. It is a better challenge than a's name, which may be rather easy to guess.

Logician: Let us see. So the first message that a sends to b consists of a pair

of a nonce n_a and the name of a, encrypted with the public key of b. These public keys are there for grabs, remember. But only b can decrypt this and get hold of n_a and a. Now b creates his own nonce n_b, and sends the pair of the two nonces n_a and n_b back to a, encrypted with a's public key.

Philosopher: Presumably, this is meant to prove to a that this message is indeed from b. Only a can decrypt this, so the final message, where a sends n_b back to b encrypted with b's public key, is supposed to prove to b that he is indeed talking to a. But how can we be certain that the protocol is secure?

Security Analyst: This is the interesting but tricky field of verification of security protocols. For example, the Needham-Schroeder protocol was first proved secure using a special "logic of authentication", but later it was found to contain a security hole after all.

Logician: (Looks challenged, and thinks out loud) But then this logical analysis did not adequately cover the essentials of the protocol. I mean, it is not so clear what verification of the protocol means. What, exactly, are the properties we should check? Informally, the protocol should ensure that a and b know with whom they have been interacting. I guess this means that they should both know the values of n_a and n_b. Moreover, it should be known to a and b that no one else knows the values of n_a and n_b. I suppose one would need epistemic logic to check such properties...

Security Analyst: (Interrupts) The attack on the Needham-Schroeder protocol was detected ten years ago, using process-theoretic tools [5]. The protocol itself is from as early as [6]. The "correctness proof" was given in [1], in the paper which introduced the logic of authentication, which was baptized *BAN-logic* after the authors. To their credit one should say that, even if their proof was flawed, their paper initiated the now active field of verification of security protocols. By the way, Needham himself once described security protocols as "three line programs that people still manage to get wrong". Well, he was right in this particular case.

Logician: I am curious, both about the logic and about the attack.

Security Analyst: About the logic... You'll be interested to learn that it reasons about *beliefs* in a very abstract way, by specifying inference rules for that. But I would say that it has no sensible semantics.

Logician: Well, that makes it hard to talk about soundness and completeness, I guess.

Security Analyst: Lowe's attack, however, is easily explained. Assume a initiates a session with c, whom she trusts. So a sends $\{n_a, a\}_{\mathrm{PK}_c}$. Instead of responding as specified by the protocol, c passes the message on to b, encoded with b's public key. b now thinks he is talking to a and sends $\{n_a, n_b\}_{\mathrm{PK}_a}$ back. This is intercepted by c and forwarded to a later. Now a still thinks she is running the protocol in interaction with c, so she responds with $\{n_b\}_{\mathrm{PK}_c}$. To conclude the protocol with b, c decrypts this message and encrypts it with b's public key to forward the nonce to b. So b ends up with the mistaken belief that he is talking to a, while he is in fact talking to c: *(writes on the whiteboard)*

Message 1	$a \to c:$	$\{n_a, a\}_{\mathrm{PK}_c}$
Message 1'	$c(a) \to b:$	$\{n_a, a\}_{\mathrm{PK}_b}$
Message 2	$b \to c(a):$	$\{n_a, n_b\}_{\mathrm{PK}_a}$
Message 2'	$c \to a:$	$\{n_a, n_b\}_{\mathrm{PK}_a}$
Message 3	$a \to c:$	$\{n_b\}_{\mathrm{PK}_c}$
Message 3'	$c(a) \to b:$	$\{n_b\}_{\mathrm{PK}_b}$

When I write $c(a)$ I mean c masquerading as a. This compromises the protocol, for it is clear that after this session b mistakenly believes that b and a are the only ones who know n_a and n_b. Also, a has the mistaken belief that a and c are the only ones knowing these nonces.

Computer Scientist: This is all going much too fast. Can we go back to the very first line of the Needham-Schroeder protocol, please? (*He points at the first line of the original protocol, still on the whiteboard.*)

$$\text{Message 1} \quad a \to b: \quad \{n_a, a\}_{PK_b}$$

This says that agent a sends a message to b consisting of the nonce n_a and the name a, but encrypted by b's public key. Now a sends to b...

Security Analyst: That is not the whole story. In the analysis of security protocols we always assume that there could be some bad guys who are trying to gain information or spoil communication. In an actual run according to the protocol, an eavesdropper might also receive this message. But assuming that the eavesdropper does not have b's private key, he does not really learn anything from it.

Cognitive Scientist: "Not really learn anything" is too vague for me.

Philosopher: And what do you mean by an actual run "according to the protocol"?

Security Analyst: (*Sighs*) OK. *(Turns to Cognitive Scientist.)* "Not really learn anything" means that if the eavesdropper tries to guess the content of the decrypted message, there is no algorithm using the encrypted message that is significantly faster than an algorithm that just performs random guesses. Please don't ask me to write down the definitions, they are long, technical, and boring.

Logician: But wait. I think the eavesdropper does learn something! At least he now knows that there was a message being sent intended for someone.

Security Analyst: Yes, but does it mean anything useful to him?

Logician: Well, that depends on the circumstances. He may learn from it that a run is going on, for instance. Maybe it is useful for him to reason about other stuff in the remaining run. I feel dynamic epistemic logic could help here to express such subtleties, and then it may become visible whether it can be useful... *(Smiles and drifts away in thoughts.)*

Philosopher: Interesting indeed, but what about my question? What do you mean by "according to the protocol"?

Security Analyst: Ah, I meant that the pattern of the actual actions matches the protocol specification.

Philosopher: Ahem, that is still not completely clear. You specified action patterns from an outsider's perspective. But what about the agents' perspective? How do they know what they should do to arrive at a run "according to the protocol"?

Computer Scientist: Maybe we should require that the protocol specification also contain the preconditions for agents to do a certain action according to the protocol. Then we know what are the possible runs.

Security Analyst: I see your point. Actually, for traditional security protocol verification, using model checking, a run generator—which produces all the possible runs according to the protocol specifications—is crucial. The preconditions for the actions are always implicitly assumed to arise from some kind of pattern matching. For example, as presented in [2], the author assumes that trusted agent b will send message $\{V, n_b\}_{\mathrm{PK}_a}$ whenever he reads message $\{V, a\}_{\mathrm{PK}_b}$ according to the Needham-Schroeder protocol. Here V is just a

variable which can be instantiated in a specific run.

Cognitive Scientist: Okay, that sounds reasonable. After all, the agents themselves can't see that V is indeed the nonce n_a generated by a. It shouldn't be in the preconditions of the actions. I would expect names of agents would sometimes be variables as well.

Security Analyst: Yes. In fact we call the names "roles" in the protocol specifications since in the actual runs any number of agents can be involved in multiple sessions. One particular agent can be acting the a role in one session but the b role in another session.

Computer Scientist: Yet another complication.

Security Analyst: I am afraid it cannot be helped. However, the variables we introduced give us executable protocol specifications for every agent. The possible runs—or the possible action sequences—are the sequences in which every action's precondition was satisfied after the execution of the previous action.

Logician: Let us go back to the attack. I am still not fully satisfied there. For example, in the description of the attack, the agents look a bit gullible. If I were b, I wouldn't have believed that what I just received was straight from a. And if I had seen something going on between a and c, I might even have discontinued the run. In other words, if I knew about the possible attack, I would be suspicious upon receiving the message $\{a, n_a\}_{\mathrm{PK}_b}$. It seems that you assume all the agents have so much faith in the protocol that they suspend their own judgement.

Security Analyst: I see what you mean. But maybe you should bear in mind that an agent need not be human. Think of the agents as communicating processors. They don't have reasoning power, or a will of their own.

Philosopher: (towards the logician) If you give the agents unlimited reasoning power, as you seem to suggest, they turn into a kind of perfect logicians. Which means that they will be able to find out about possible attacks beforehand, by analyzing the protocol as you would analyze it yourself. Or in the way Lowe analyzed the Needham-Schroeder protocol. Just imagine. You then assume that if there exists an attack, they will discover it, and then they refuse to conduct any run: "We know we are under threat of attack". And then all security analysts will lose their jobs... *(smiles)*

Logician: But at least we have to make explicit what the reasoning and observation powers of the agents are. Maybe for different situations we need different assumptions.

Computer Scientist: It seems to me that our discussion has revealed quite a list of tacit assumptions. Why don't we try and make a list?

Cognitive Scientist: Yeah!

Security Analyst: Well, we did not yet discuss assumptions about the bad guys. A model where such a bad guy is assumed to exist is called an *intruder model* or *threat model*. An intruder model that is well-known is the so-called Dolev-Yao model [3]. In this model it is assumed that the intruder has complete control over the communication channel. That is, he can intercept every message, delete messages at will, and insert messages into the communication channel. Of course, the model is supposed to be realistic about what intruders can do and cannot do. It is assumed that the intruder cannot work magic. An intruder can only insert messages that can be construed in polynomial time on the basis of the information he already possesses.

Philosopher: So, if I look at the first message of the Needham-Schroeder protocol, the $a \rightarrow b$ only means that it prescribes a to send some message, but there is no guarantee that b gets the message, nor, if b gets some message, that he can be sure that the message he receives originates from a. Is that correct?

Security Analyst: Yes. But we do assume that the encryption works as it should: the message sent there can only be decrypted by agent b. Recall that we treat the cryptographic primitives as perfect black boxes [9]. Thus, hashes, encryption, decryption and what-have-you all exist, and work as they should. We don't worry about the mathematical foundations of their existence, or about details of their implementation. By the way, this does not mean that cryptographic primitives are not interesting: For example the definition of what a cryptographic hash should do has been in a constant flux over the last twenty years. I won't go into details here, but I recommend reading [10, chapter 3] for a nice wrap-up of this history.

Cognitive Scientist: Hello? I thought you were all doing exact sciences. Now I am surprised how much is still under debate, and how dubious your verification methods are. Like this BAN-logic, which has no semantics and which proves flawed protocols correct. Deep maths is not a guarantee for maturity, or is it?

By the looks of it, your discipline is still in its infancy. Just like ours, in fact.

Computer Scientist: We're not even finished yet with the list of assumptions. In the Needham-Schroeder protocol, the claim of authentication relies on the fact that the nonces n_a and n_b are only known to a and b, respectively. But why wouldn't a or b disclose these nonces?

Security Analyst: Well, they don't. And yes, that is another assumption for the trusted agents. So the claim is in fact conditional if we spell out the types of the agents as follows: If both agent a and b are trusted then they can identify each other after any run of the protocol.

Logician: The list of assumptions keeps growing. What worries me more is that the list of quantifications is also rapidly growing.

Philosopher: Yes, indeed, it is.

Cognitive Scientist: What do you mean? Quantifications over what?

Logician: Well, first, if a principal sends an encrypted message, everybody who does not possess the decryption key deems any other message possible. Second, the bad guy or the so-called intruder of this Dolev-Yao model can do anything anytime. He can insert messages, almost any message, and he can delete messages that are sent.

Security Analyst: Yes, we have to be careful to keep our model manageable. To answer your first point: we can assume trusted agents that only do pattern matching to pick up the right messages. And I do agree with your remark about quantifications: if we need to take into account all the possible behaviors of non-trusted agents, the model grows huge! Another thing to bear in mind, by the way, is that the fewer assumptions you make, the more general will be your correctness proof...

Philosopher: Now let's finally make a list of all the aspects of the modeling we discussed.

Logician: Well, this is just preliminary, of course (*Stands up and walks to the whiteboard and draws*):

- **Protocol description**
  - executable protocol specification for each agent (who should do what under what preconditions);

  - initial distribution of information and kept secrets (e.g. the information about the names of participants, the public/private keys);
  - requirements (what facts should be true at the end of the protocol or even in the middle stage of a run of the protocol?)

- **Assumptions**
  - primitives (what cryptographic primitives are used?)
  - intruder model (what can the intruder do?)
  - trusted agent model (how do they behave, reason and observe?)
  - communication model (how the messages are sent and received, related to agents' observations.)

Cognitive Scientist: Well, you are the logician, what do you make of all these questions?

Logician: The challenge is to find a logical system that fits these situations like a glove. Actually, I do have some ideas for such a logic already...

Philosopher: That's great! But for now I think we should call it a day. May I invite you all to a delicious dinner at NIAS?

References

[1] Michael Burrows, Martín Abadi, and Roger Needham. A logic of authentication. *Proceedings of the Royal Society of London, Series A, Mathematical and Physical Sciences*, 426(1871):233–271, December 1989.

[2] C.J.F. Cremers. *Scyther — Semantics and Verification of Security Protocols*. PhD thesis, Technical University Eindhoven, 2006.

[3] Danny Dolev and Andrew C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, March 1983.

[4] K.A. Ericsson and H.A. Simon. *Protocol analysis: Verbal Reports as Data*. MIT Press, Cambridge, MA, 1984.

[5] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Proceedings of TACAS*, volume 1055 of *LNCS*, 1996.

[6] R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12), 1978.

[7] A. Newell and H.A Simon. *Human Problem Solving*. Prentice-Hall, Englewood Cliffs, NJ, 1972.

[8] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[9] Claus Peter Schnorr. The black-box model for cryptographic primitives. *Journal of Cryptology*, 11(2):125–140, March 1998.

[10] Wouter Teepe. *Reconciling Information Exchange and Confidentiality — A Formal Approach*. PhD thesis, Rijksuniversiteit Groningen, 2007.