Formal Methods for Security Protocols: Three Examples of the Black-Box Approach

C.J.F. Cremers [1], S. Mauw [1] & E.P. de Vink [1,2]

[1] Department of Mathematics and Computer Science
Technische Universiteit Eindhoven
{ccremers,sjouke,evink}@win.tue.nl

[2] LIACS, Universiteit Leiden

Abstract Security protocols are hard to design, even under the assumption of perfect cryptography. With the 'classical' Needham-Schröder protocol as leading example, three so-called black-box approaches to the formal verification of security protocols are sketched. *BAN-logic* is a light-weight method under the assumption of honest agents and a passive intruder. The *Casper/FDR* approach translates a high-level description of a security protocol, together with its security requirements and a particular instantiation into CSP, that can be machine-verified using the FDR model checker. In this approach the intruder is in control of the network and is allowed to participate as one of the agents. The same holds for the *strand space* approach. By focusing on the causal dependency of actions and message passing, one aims to prove security properties of general class of protocols instantiations at the same time.

Keywords Security protocols, formal verification, BAN-logic, Casper, FDR, strand spaces

1 Introduction

Security protocols are notoriously known to be difficult to get right. Many protocols proposed in the literature and many protocols exploited in practice turned out to be flawed, or their well-functioning was found to be based on implicit assumptions. Since the late eighties various approaches have been put forward for the formal verification of security protocols to overcome the problems of faulty implementations and hidden requirements.

In this contribution we will focus on three methods for the formal verification of security protocols: BAN-logic, the Casper/FDR-approach and strand spaces. These methods, among others, have in common that they abstract away from cryptographic details. Instead they assume that suitable cryptographic primitives like encryption and decryption, digital signing and verification of signatures are given and are cryptographically safe. So, considerations of weak RSA-moduli or smooth primes numbers, for example, are not part of the investigations. Of course, these aspects are relevant to the safety of a security protocol. No application should be packed and shipped before the crypto-analytic weaknesses of the various underlying algorithms have been carefully pondered about and the subsequent risks have been weighted against various odds and evens. However, even when abstracting away from the cryptographic issues by starting from their complete safety, security protocols can still go to wreck.

A classical example in the short history of security protocols is the protocol due to Needham and Schröder. One of the aims of the protocol run by two, possibly unacquainted parties, Alice and Bob say, is that they can authenticate each other on the basis of their public keys. The protocol presumes a public/private key infrastructure, which means that every agent has a secret private key and a corresponding public key which is known to all other agents. The protocol is explained by means of a message sequence chart in Figure 1. Encryption of a message m by key K is denoted as $\{m\}_K$. We assume Alice and Bob to hold or be

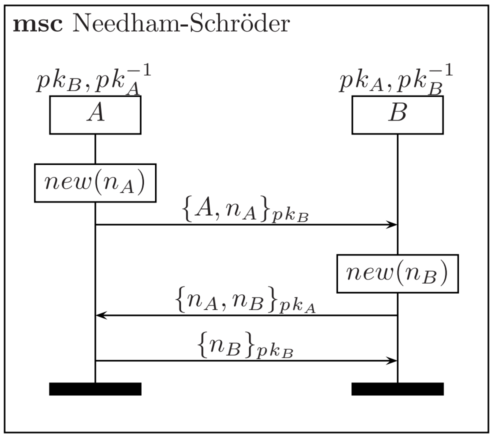

Figure 1: Message sequence chart of the Needham-Schröder protocol

able to look up the public keys of each other. Alice creates a new so-called nonce, a fresh random number n_A. Then Alice sends an initiating message to Bob consisting of her name and the nonce, both encrypted with the public key pk_B of Bob. In this way, Alice can make sure that only Bob can unpack the message and learn the nonce n_A. Bob, assumed to be in the possession of the private key pk_B^{-1} corresponding to the public key pk_B, decrypts the incoming message and retrieves two components, viz. the nonce n_A and the identity of Alice. Knowing which protocol to follow, Bob creates a fresh nonce of his own and replies to Alice by adding his nonce to the one of Alice, both encrypted with the public key pk_A of Alice. The fact that he replies to the previous message in the correct way serves to give evidence to Alice that he is really Bob. Alice decrypts the message with her private key, checks that it contains her original nonce n_A and finds a new nonce, apparently from Bob. She responds to Bob by sending out this latter nonce encrypted under the public key of Bob. Thus, she proves her identity to Bob. So, at the end of the protocol both Alice and Bob are convinced that they are engaged in a protocol run with each other and that one nonce is coming from Bob and the other is coming from Alice.

In the remainder of this paper we want, by means of illustration of the various formal approaches, to verify a single security property of the Needham-Schröder protocol only. It will turn out that the underlying security model, that is, the capabilities of the intruder, is crucial to the proper interpretation of the merits of the protocol. We start off with the method set out by Burrows, Abadi and Needham.

Note The reported here is no original research of the authors. For activities of the Eindhoven Computer System Security group see `http://www.win.tue.nl/~eccs`.

2 BAN-logic

BAN-logic, named after its inventors, is a first-order modal logic with modalities of observation '$\triangleleft$', '$\mid\!\sim$' and '$\vdash$' and a modality of belief '$\mid\!\equiv$'. In the presentation here, we restrict to predicates expressing possession '$\ni$', key binding 'pk', freshness '$fresh$' and sharing of secrets '$secret$'.

Within the logic, statements can be made about a current run of a protocol in an imperfect network eavesdropped by an intruder. The intuition of the formula $P \lhd X$, read as 'P is told X', is that agent P receives X either as an entire message or as a submessage that he can distill. A distinction is made between the present and the past. The current run takes place in the present; all other runs have taken place in the past. For example, the formula $P \mid\!\sim \phi$ reads as 'P once said ϕ, either now or earlier', whereas the formula $P \vdash \phi$ states that 'P uttered ϕ in the current run'. The operator '$\models$' is used for constructs as $P \models \phi$, meaning that P is entitled to believe that ϕ holds. Formulas that are derivable in the logic are valid statements about the current run of the protocol irrespective of the residuals of possible earlier runs. In general, we want to prove statements of the form $P \models Q \vdash \phi$ or $P \models Q \models \psi$, i.e. that agent P is entitled to believe that agent Q said something in the current run of the protocol or that agent P legitimately believes that agent Q is at present entitled to believe something else.

Other ingredients that we consider in the BAN-analysis of the Needham-Schröder protocol: freshness predicates $fresh(X)$ stating that a message has been generated in the present, hence not in the past; possession formula $P \ni X$ for some key, nonce or name X expressing that agent P is in possession of or knows that key, nonce or name; public-key claims $pk(P, K)$ that the public key K is associated with agent P; statements of shared secrets $secret(X, P, Q)$ saying that nonce or key X is a secret shared by agents P and Q.

As deduction rules we will focus on the following three rules:

$$\text{(Dec)} \qquad \frac{P \lhd \{X\}_K, \quad P \ni K^{-1}}{P \lhd X} \qquad\qquad \text{(SS)} \qquad \frac{P \lhd X, Y, \quad P \models secret(X, P, Q)}{P \models Q \mid\!\sim X, Y}$$

$$\text{(Say)} \qquad \frac{P \models Q \mid\!\sim X, \quad P \models fresh(X)}{P \models Q \vdash X}$$

The decryption rule (Dec) states that if agent P has received a message X encrypted with the key K and P is in possession of the decryption key corresponding to K, P has received X. The shared secret rule (SS) expresses that if agent P is told the message consisting of X and Y and P believes X to be a secret it shares with agent Q, then the message X, Y has been sent (either now or in the past) by agent Q. Note that the soundness of the rule assumes that the intruder is not capable of manipulating and sending out messages himself. Otherwise, the intruder simply replaces the component Y by his favorite component Z, making agent P to believe that agent Q once sent X, Z. The presence of fresh items pinpoints the sending of messages at the present as captured by the rule (Say): if agent P believes that agent Q once said message X and P also believes that X has been generated in the current run, then P believes that Q uttered the message X in fact in the current run.

The security goal of the Needham-Schröder protocol that we are focusing on can be rephrased in terms of the BAN-logic as $B \models A \vdash n_B$, i.e. Bob believes that Alice said n_A, in the last step of the protocol. From this it follows that Alice was engaged in the protocol with Bob. A proof in BAN-logic that the protocol supports this runs as follows: First we list the initial knowledge. We assume $A \models pk(B, pk_B)$, $A \ni pk_A^{-1}$, i.e. A believes that the public-key of B is the key pk_B and that A possesses his private key. Likewise for Bob, $B \models pk(A, pk_A)$, $B \ni pk_B^{-1}$. After the first action of the Needham-Schröder protocol we have that $A \models fresh(n_A)$ as Alice has generated it herself in the current run. Next, after the first message sent out by Alice, we obtain $A \models secret(n_A, A, B)$ and $B \lhd \{A, n_A\}_{pk_B}$, meaning, respectively, that Alice believes that she shares the secret n_A with Bob as she has sent it under Bob's public key, and, that Bob receives the name A together with the nonce n_A encrypted with his public key. The formulas $A \models fresh(n_A)$, $A \models secret(n_A, A, B)$ and $B \lhd \{A, n_A\}_{pk_B}$ are typical of what 'axioms' can be obtained directly from the protocol: nonce or key generation, exchange of secrets, receipt of message. Using these kinds of facts we have, for example, the following derivation in Figure 2.

(1)	$B \lhd \{n_B\}_{pk_B}$	(message 3)
(2)	$B \ni pk_B^{-1}$	(assumption)
(3)	$B \lhd n_B$	(1, 2, PK)
(4)	$B \models secret(n_B, A, B)$	(message 2)
(5)	$B \models A \mid\!\sim n_B$	(3, 4, SS)
(6)	$B \models fresh(n_B)$	(action 2)
(7)	$B \models A \vdash n_B$	(5, 6, Say)

Figure 2: Proof in BAN-logic of $B \models A \vdash n_B$

In the proof we have exploited all the rules mentioned above, initial knowledge of the agent A, and results of the preceding action and message. The conclusion is that Alice rightly believes that Bob is involved in the protocol. Stated otherwise, Bob is authenticated by Alice. As touched upon in the discussion of the shared secret rule, the underlying model assumes that A and B are honest, i.e. behave according to the protocol, and that the intruder can overhear but not disturb the sending and receiving of messages. In the next section we will reconsider this and assume the intruder to be substantially more powerful.

3 Casper/FDR

The Needham-Schröder protocol was published in 1978. It came as a surprise that Lowe found a weakness in 1996 only, as the protocol was considered to be safe. However, in the late seventies computer networks were closed networks. This had changed completely in the mid nineties. The setting of computer networks had changed from closed to open networks. As a consequence the capabilities that should be attributed to an intruder had grown, and, unfortunately, the BAN-analysis of the Needham-Schröder protocol is not valid in this context. it is relatively simple to intercept, alter and spoof ethernet or Internet traffic.

Nowadays the standard setting is that agents are not necessarily honest and that the network is under control of the intruder who can block, inject and invent messages. However, cryptography is assumed to be perfect. The intruder, and all others involved, can only see the contents of an encrypted message if they are in possession of the corresponding decryption key. This is the black box assumption.

The attack of Lowe is a so-called man-in-the-middle attack, in which the intruder fools Bob to think that he is talking to Alice where he is in fact talking to the intruder. And what is worse, Bob might subsequently reveal secrets to the intruder that are only to the discretion of Alice and Bob. The attack assumes that the intruder has a public-key private-key pair and an identity I of its own in the network, and that Alice initiates a session with I. See Figure 3 for the message sequence chart. Note that no cryptographic tricks have been used. Simply by

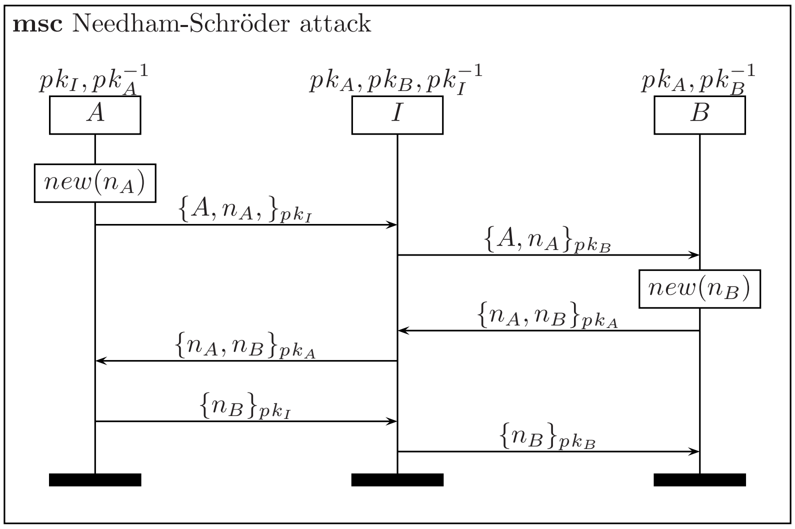

Figure 3: Man-in-the-middle attack on the Needham-Schröder protocol

redirecting messages while running two runs of the protocol in parallel, the intruder manages to impersonate himself to Bob.

Finding attacks mechanically with a tool is one of the appealing characteristics of the Casper/FDR approach. The Casper tool, developed by Lowe, translates a high-level description of a protocol into CSP, the process algebra of Communicating Sequential Processes developed by Hoare and co-workers. The CSP-program comprises various processes representing the agents involved in the protocol that run in parallel with an intruder that controls the network. Additionally the CSP-program contains the specification of the security requirements the protocol should meet.

The CSP-program is fed in to the FDR-tool, an industrial-size modelchecker for CSP. The modelchecker checks for all the traces obtained from interleaving of the possible actions of the agents and intruder (blocking, forwarding, and spoofing of messages) that they fit the format described by the security requirements. So, FDR verifies that the implementation is a trace refinement of its specification. If not, a counter example is found. Casper can translate the FDR-output in human-readable form, yielding a blueprint for an attack on the protocol. This Casper-FDR-Casper chain can be automated through the use of the CasperFDR script.

A Casper input file, see Figure 4, consists of a general part (variable and process declaration, protocol description and specification of requirement) and a specific part that instantiates the general part and specifies the intruder knowledge. The free variables section contains the functions `pk` and `sk` that associate a public key and a secrete or private key with each agent. The inverse keys declaration specifies that for any particular agent these keys are the inverses of each other. The processes section introduces an initiator process with the agent `A` and the nonce `nA` as parameters. Moreover, the process has all public keys and the private key of the agent `A` at its disposal. Likewise, a responder process is declared. The protocol section gives the protocol itself. In the step '`0.   -> A : B`' the environment, say the user of the machine `A`, is providing the name `B`. The security requirements can be found in the specification section. Here we have only have one agreement requirement, stating that the

```
-- Needham Schroeder

#Free variables
A, B : Agent
nA, nB : Nonce
pk : Agent -> PublicKey
sk : Agent -> SecretKey
InverseKeys = (pk, sk)

#Processes
INITIATOR(A,nA) knows pk, sk(A)
RESPONDER(B,nB) knows pk, sk(B)

#Protocol description
0.    -> A : B
1.  A -> B : {A, nA}{pk(B)}
2.  B -> A : {nA, nB}{pk(A)}
3.  A -> B : {nB}{pk(B)}

#Specification
Agreement(B,A,[nA,nB])

#Actual variables
Alice, Bob, Ivey : Agent
Na, Nb, Ni : Nonce

#Functions
symbolic pk, sk

#System
INITIATOR(Alice, Na)
RESPONDER(Bob, Nb)

#Intruder Information
Intruder = Ivey
IntruderKnowledge =
  {Alice, Bob, Ivey, Ni, pk, sk(Ivey)}
```

Figure 4: Protocol description of Needham-Schröder in Casper

agent `A` is authenticated to agent `B` and that they agree on the values of the nonces `nA` and `nB`.

In the instantiation of the general part we provide, in the actual variables section, besides two honest agents `Alice` and `Bob` also a third agent `Ivey` that will play the role of the intruder. We will not use specific properties of the `pk` and `sk` functions apart from being each others inverse. The systems section specifies the processes that comprise the system, in this case `Alice` as initiator and `Bob` as responder. The process for the intruder `Ivey` is implicit. The process is inserted by Casper into the CSP file. The intruder can do whatever he likes based on the knowledge he has gathered so far.

The Casper compiler checks that the protocol is feasible, i.e. that parties are able to send out the messages as specified. This is related to the active role of the intruder. The intruder can inject any message that it can construct at that time. Therefore, a mechanism should be in place that controls the nondeterminism of the intruder. It is also important to eliminate improper traces as soon as possible because of the state space explosion problem.

When the protocol specification for the Needham-Schröder protocol as given in Figure 4 is processed through the CasperFDR script, a violation to the security requirement is found. The particular trace found by FDR is interpreted by Casper, see Figure 5. The intruder plays both the role of Ivey and of Alice, thereby successfully masquerading its identity to Bob, who believes he is dealing with Alice.

With the Casper/FDR machinery available one can easily play with the protocol and try to find an improvement of the Needham-Schröder protocol that is not amenable to the man-in-the-middle attack. This way, the solution proposed by Lowe himself can be recovered. The message sequence chart of the improved Needham-Schröder-Lowe protocol is given in Figure 6. The repair consists of including the responder's identity in the second message. For this new protocol and the particular instance of the system the modelchecker does not discover violating traces any more.

```
Attack found:
Top level trace:
  Alice believes she is running the protocol,
  taking role INITIATOR, with Ivey, using data items Na, Nb
  Bob believes he has completed a run of the protocol,
  taking role RESPONDER, with Alice, using data items Na, Nb

System level:
0.          ->  Alice  : Ivey
1.  Alice  -> I_Ivey  : {Alice, Na}{pk(Ivey)}
1. I_Alice ->   Bob   : {Alice, Na}{pk(Bob)}
2.   Bob   -> I_Alice : {Na, Nb}{pk(Alice)}
2. I_Ivey  ->  Alice  : {Na, Nb}{pk(Alice)}
3.  Alice  -> I_Ivey  : {Nb}{pk(Ivey)}
3. I_Alice ->   Bob   : {Nb}{pk(Bob)}
```

Figure 5: CasperFDR output indicating the man-in-the-middle attack

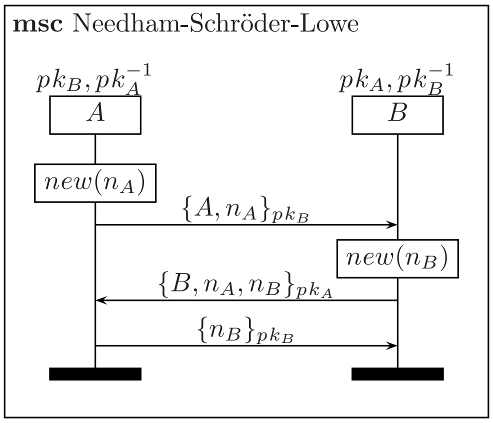

Figure 6: Message sequence chart of the Needham-Schröder-Lowe protocol

Note that in the system instantiation in Figure 4 only one initiator and one responder are specified. This is an important restriction of this approach. If we want to analyze the situation with two protocols running in parallel, say between Alice and Bob and between Alice and Claire, we would write, e.g.,

```
INITIATOR(Alice, Na1)
RESPONDER(Bob, Nb)
INITIATOR(Alice, Na2)
RESPONDER(Claire, Nc)
```

The guarantee of the Casper/FDR analysis restricts to the particular case as specified (with a strong intruder), whereas the BAN-analysis quantifies over all possible past behaviors (but with a weaker intruder).

In general, machine tools can only analyze a finite search space. Casper enforces this by analyzing a protocol for a specific number of protocol runs. Another restriction of the approach is that Casper comes with a number of predefined security requirements. To correctly model a protocol, a thorough understanding of the subtleties of the security requirements in Casper is needed.

4 Strand spaces

The last method of analyzing security protocols that we discuss here is the so-called strand spaces approach as advocated by Thayer Fábrega, Herzog and Guttman. Strand spaces are also based on the so-called Dolev-Yao model where the intruder is a first-class citizen and, moreover, in control of the network. In the Casper/FDR approach a specific system instantiation is considered in isolation. In the strand spaces approach one seeks, instead, to prove properties of arbitrary combinations of protocols running at the same time.

In the strand space approach agent activity is modelled as a number of sequential threads put in parallel. These threads are the strands. The nodes on the strands are the actions the agent performs. The sequence of actions along the strand is referred to as its trace. The nodes on a single strand are causally related. Between the strands there is in general the sending and receiving of messages. So, also between a sending node and a receiving node there is a causal relationship. Now, a strand space is a collection of strands reflecting the activity of honest agents involved in one or more protocols together with a number of strands of the intruder. The nodes on all the strands together form a partial order when endowed with the causality relation induced by the sequentiality on a single strand and the sending and receiving of messages. The strands of the intruder are of a restricted form to be given in a minute.

In Figure 7 a strand space is depicted showing the attack on the Needham-Schröder protocol as described in Section 3. The double lines are strands, the nodes of which are executed in top to bottom order; the isolated bullets are single node strands. The signs attached to nodes indicate whether the node is an input or an output node. The minus sign of an input node indicates that this node represents the receipt of a message. A plus sign of an output node indicates that this node represents the sending of a message or that the particular message belongs to the knowledge of the agent. This latter situation is for example the case in the single node strands $+pk_I^{-1}$ and $+pk_B$. The arrows connect an input node with a corresponding output node. In a strand space, by definition, each input node must be connected to a unique output node. There are no messages of 'outer space'.

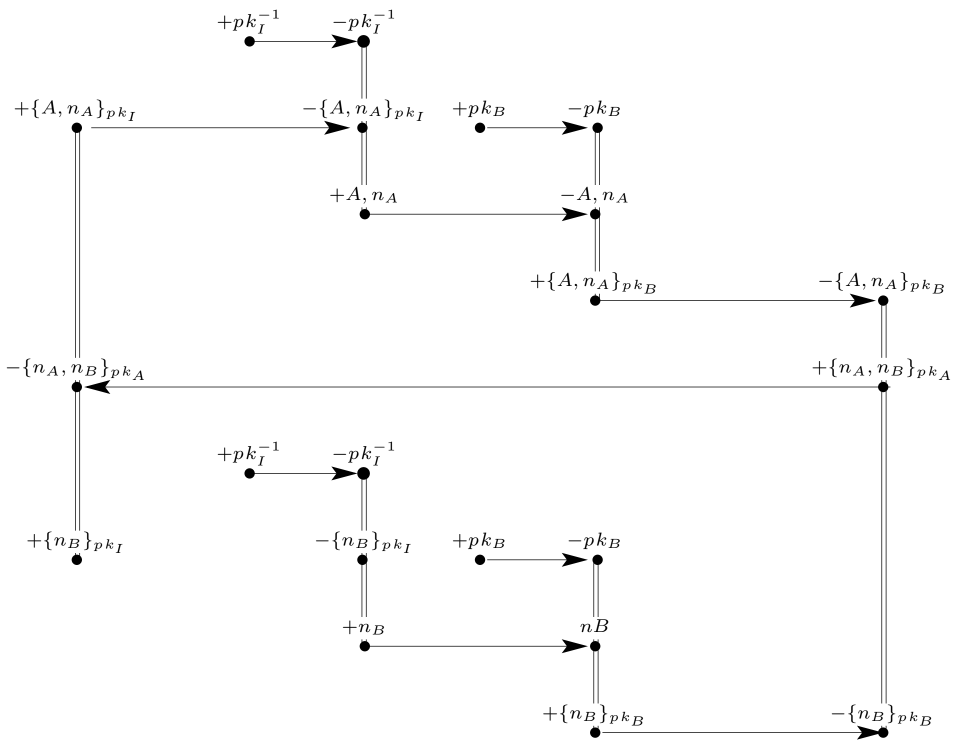

Figure 7: Needham-Schröder strand space

In Figure 7, the left-most and right-most strand are 'regular' strands expressing the activity of honest users, where as the remaining six strands in the middle are 'penetrator' strands. They represent intruder activity. The regular strands are given by the prescribed behaviour of the agents. The penetrator strands are strands with the following traces: text message or knowledge $\langle +t \rangle$, flushing $\langle -x \rangle$, tee $\langle -x, +x, +x \rangle$, concatenation $\langle -x, -y, +xy \rangle$, separation $\langle -xy, +x, +y \rangle$, encryption $\langle -k, -x, +\{x\}_k \rangle$, and decryption $\langle -k^{-1}, -\{x\}_k, +x \rangle$. These strands express the various capabilities of the intruder. The intruder can output any text or message it knows. The intruder can absorb or block messages via the flushing strand and it can make copies using the tee strand. Combination of messages or separation of messages is done with the corresponding strand. The encryption strand can be read as, given a key k and given a message x the intruder is capable of sending the message $\{x\}_k$. Similarly for a decryption strand, given the inverse key k^{-1} and an encrypted message $\{x\}_k$, the intruder can recover the contents x.

The attractive characteristic of the strand space approach is that properties of a whole class of protocol instantiations can be proven, instead of one instance at the time. For example, for the improved Needham-Schröder-Lowe protocol, one can prove on the basis of the causal structure of the strand spaces involved that if a responder can finish his part of the protocol then, under reasonable conditions, there must have been an agent running the protocol with the same parameters as an initiator.

More precisely, let $\mathit{Init}[A, B, n_A, n_B]$ be a strand of the format

$$\langle +\{A, n_A\}_{pk_B}, -\{B, n_A, n_B\}_{pk_A}, +\{n_B\}_{pk_B} \rangle$$

and $Resp[A, B, n_A, n_B]$ the complementary strand of the form

$$\langle -\{A, n_A\}_{pk_B}, +\{B, n_A, n_B\}_{pk_A}, -\{n_B\}_{pk_B} \rangle$$

Let Σ be any strand space based on initiator strands and responder strands as described, with A, B ranging over agent names and n_A, n_B ranging over nonces, and any number of penetrator strands of the seven formats given. If Σ contains a strand of the form $Resp[A, B, n_A, n_B]$, Σ does not have a knowledge strand $\langle +pk_A^{-1} \rangle$ and there is no other strand at which the nonce n_B is created as well, then Σ contains necessarily an initiator strand $Init[A, B, n_A, n_B]$. I.e., if the private key of the initiator A is not revealed and there is no confusion about the nonce n_B used by the responder B, then, upon completion of the protocol, the responder can be sure that the initiator has finished her part of the protocol using the nonce n_A.

The proof of this authentication property is based on a case analysis of the possible penetrator strands. The cutting down of the intruder behavior to specific atomic forms is instrumental to this. No specific assumptions, except for the one stated, are needed on the intruders actions. Starting from an input node of the responder the information flows, as it were, backwards into the intruder strands, thereby instantiating the parameters of the strands. Then, based on the secrecy of the private keys and the form of the actions of the initiator and responder strands all 'bad' situations in the strand space are ruled out. It should be noted that the proof indeed makes use of the form of the adapted message $\{B, n_A, n_B\}_{pk_A}$, confirming that the proof does not apply to the original Needham-Schröder protocol.

5 Conclusions

Above we have given a brief outline of three complementary approaches to formal verification of security protocols. Each approach has different limitations. In all the three methods described cryptographic details were abstracted away. It is simply assumed that an agent, including the intruder, is not capable of decrypting a message without holding the proper key. This is what is referred to as black-box cryptography.

First, the logical approach of Burrows, Abadi and Needham, that is valid in a setting of honest agents, in which the intruder is only capable of overhearing the messages. There are two drawbacks to this method. Defining a semantics for the logic is still a topic of ongoing research. In particular, a semantics combining Krypke-structures and action updates would be attractive to have. However, the underlying intruder model is significantly weaker than the common Dolev-Yao model.

Second, the modelchecker based approach of Casper/FDR in which the front-end Casper compiles a high-level description in a CSP program, that can be proven or disproven to be a trace-refinement of its specification, with the FDR-tool. Casper provides a powerful intruder model and a user-friendly interface, but choosing the right Casper requirements requires thorough knowledge of the subject. The number of protocols runs and specific instantiations must be explicitly modeled in the specifications to avoid an infinite search space.

Third, the strand spaces approach where agent activities are seen as a partial order and the intruder behavior is fragmented into basic steps, allowing for a causality based reasoning. Although mathematically appealing, the state-of-the-art is rather ad-hoc. No general method of evaluating security prperties has been emerged so far. Directly related to this is the lack of machine support for the analyis of security protocol in the strand space approach.

We are currently researching security protocol models that do not have these restrictions. Whether an approach is possible that includes the strengths of the methods mentioned here, whilst avoiding their weaknesses, is still an open question.

6 Bibliographic remarks

In this overview we have been short and, for the sake of presentation, been imprecise at certain points. The reader can consult the following literature for more detail: In [2] an authentication logic is discussed that has later been baptized BAN-logic by others. In the accessible [5] an early variation of BAN-logic is presented. The formal semantics of BAN-logic, based on so-called runs, is addressed in [1, 23]. Other extensions of the logic can be found in [11, 19, 17]. The application of CSP to security protocols was pioneered by Roscoe and Lowe [14, 9, 15] using the CSP model checking tool FDR and the Casper compiler [18, 10]. Using these tools the catalog of Clark and Jacob [3, 8] has been exhaustively analyzed in a student project [4]. The strand space approach is advocated by Thayer Fábrega, Herzog and Guttman in a series of papers [21, 22, 6]. The approach has various ideas in common with the inductive approach of Paulson [12, 13]. The latter uses the theorem prover Isabelle for the computer aided verification of security protocols. General text books that we would like to mention here include [20, 7, 16].

References

[1] M. Abadi and M. Tuttle. A semantics for a logic of authentication. In *Proc. Principles of Distributed Computing*, pages 201–216, Montreal, 1991.

[2] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8:18–36, 1990.

[3] John A. Clark and Jeremy L. Jacob. A survey of authentication protocol literature. Technical Report 1.0, Department of Computer Science, University of York, 1997.

[4] B. Donovan, P. Norris, and G. Lowe. Analyzing a library of security protocols using Casper and FDR, 1999.

[5] L. Gong, R. Needham, and R. Yahalom. Reasoning about belief in cryptographic protocl analysis. In *Proc. IEEE Symposium on Research in Security and Privacy*, pages 234–248, 1990.

[6] J.D. Guttman and F.J. Thayer. Authentication tests and the structure of bundles. *Theoretical Computer Science*, 238:333–380, 2002.

[7] C. Kaufman, R. Perlman, and M. Speciner. *Network security: private communication in a public world (2nd ed.).* Computer Networking and Distributed Systems. Prentice Hall PTR, 2002.

[8] Laboratoire Spécification et Vérification, ENS de Cachan. Security protocols open repository. http://www.lsv.ens-cachan.fr/spore/.

[9] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Proc. TACAS*, pages 147–166. LNCS 1055, 1996.

[10] Gavin Lowe. Casper: A compiler for the analysis of security protocols. In *Proc. 10th IEEE Computer Security Foundations Workshop*, 1997. http://web.comlab.ox.ac.uk/oucl/work/gavin.lowe/Security/Casper/index.html.

[11] P.C. van Oorschot. Extending cryptographic logics of belief to key agreement protocols. In *1st ACM Conference on Computer and Communications Security*, pages 232–243, Fairfax, Virginia, 1993.

[12] L. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6:85–128, 1998.

[13] L.C. Paulson. Proving properties of security protocols by induction. In *Proc. 10th Computer Security Foundations Workshop*, pages 70–83, Rockport, Massachusetts, 1997.

[14] A.W. Roscoe. Modelling and verifying key-exchange protocols using CSP and FDR. In *Proc. 8th IEEE Computer Security Foundations Workshop*, pages 98–107, Kenmare, 1995.

[15] P.Y.A. Ryan and S.A. Schneider. *Modelling and Analysis of Security Protocols: the CSP Approach*. Addison-Wesley, 2001. With M.H. Goldsmith, G. Lowe and A.W. Roscoe.

[16] B. Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C (2nd ed.)*. Wiley, 1995.

[17] S.G. Stubblebine and R.N. Wright. An authentication logic with formal semantics supporting synchronization, revocation, and recency. *IEEE Transaction on Software Engineering*, 28:256–285, 2002.

[18] Formal Systems. http://www.formal.demon.co.uk/FDR2.html.

[19] P.F. Syverson and P.C. van Oorschot. A unified cryptographic protocol logic. CHACS Report 5540-227, NRL, 1996.

[20] G. Tel. *Cryptografie: Beveiliging van de digitale maatschappij*. Pearson Education, 2002. In Dutch.

[21] F.J. Thayer Fábrega, J.C. Herzog, and J.D. Guttman. Strand spaces: Why is a security protocol correct? In *Proc. 1998 IEEE Symposium on Security and Privacy*, pages 66–77, Oakland, California, 1998.

[22] F.J. Thayer Fábrega, J.C. Herzog, and J.D. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7:191–230, 1999.

[23] G. Wedel and V. Kessler. Formal semantics for authentication logics. In E. Bertino, H. Kurth, and G. Martella, editors, *Proc. ESORICS'96*, pages 219–241. LNCS 1146, 1996.