Analysis of authentication protocols
Intership report

Stéphane Glondu
ENS de Cachan*

September 3, 2006

1 Introduction

Many computers are interconnected through networks, the biggest of them being Internet. Communications over these networks are ruled by *protocols*: physical networks are just wires (or radio waves) linking computers, providing channels where data can be transmitted; protocols define how data are formatted to go through these channels, and also provide additional features such as error detection and correction.

More precisely, a protocol is a sequence of messages exchanged between (two or more) *principals*. A principal can be a computer, but also—more generally—a (human) user, a process, a server, etc. Sometimes, the identities of the principals involved are important: one must be sure of the other's identity to proceed. For example, when you connect to your bank's website, you want be sure that you are really talking to your bank and that nobody is spying on you. You cannot just assume that there is an armored pipe between you and your bank: actually, your messages must go through several intermediate entities (wires, switches, routers), and each of them can be watched or controlled by a possibly malicious intruder.

A protocol resistant to intermediate malicious actions is a *security protocol*. Two main properties are targeted when designing such protocols: authentication and secrecy. These are achieved using cryptography (public-key, shared-key or both), so such protocols are also called *cryptographic protocols*. This report is mainly about verification of the security properties of cryptographic protocols. We consider especially authentication protocols, for which Clark and Jacob give a survey in [9][1].

The main target of this report is to present and compare some formal approaches to protocol verification (based on proof systems), and to present in detail a semantics for one of them and prove the corresponding soundness.

In this report, *formal* frameworks are especially considered, where some simplifying assumptions—such as perfect cryptography, uniquely interpretable

*while visiting Keio University, Tokyo, Japan (April–August 2006)

[1]This survey might seem quite old now—many analysis methods have been published since then—but it is still a good introduction to authentication protocols.

terms—allow the use of logic-related tool to prove properties. Many such methods have been proposed, following the work of Dolev and Yao [12], and Burrows, Abadi and Needham [5]. I will present in this report some of them, and use the famous Needham-Schroeder public key protocol (NSPK) [23] as an example.

Section 2 presents some formal approaches to protocol verification, namely BAN logic [5], strand spaces [14], PCL [13], MSR [8] and BPL [17]. In section 3, I give a (very short) introduction to other approaches. BPL is further investigated in section 4.

2 Formal approaches to protocol verification

2.1 The agreement property and the Needham-Schroeder public key protocol

This protocol was introduced by Needham and Schroeder in 1978 [23]. It uses public key cryptography. Here, we suppose that the principals know each other's public key. We present it in the usual—intuitive—way:

$$\begin{aligned} A &\to B : \{n_1, A\}_{K_B} \\ B &\to A : \{n_1, n_2\}_{K_A} \\ A &\to B : \{n_2\}_{K_B} \end{aligned}$$

The fist line means "A sends to B the tuple (n_1, A) encrypted by B's public key". At the end of a run, A and B are supposed to be mutually authenticated, and n_1 and n_2 are known only by them. A and B are commonly called the *initiator* and the *responder*, respectively.

More precisely, authentication can be stated in terms of *the agreement property* as explained in [14]:

> Each time a principal B completes a run of the protocol as responder using $\vec{x}$, apparently with A, then there is a unique run of the protocol with the principal A as initiator using $\vec{x}$, apparently with B.

Usually, some additional properties expressing the honesty of the participants are also added. If this and the converse property for the initiator hold, the agreement property holds.

The NSPK protocol was thought to be secure for several years, but Lowe eventually found an attack in 1995 [21]:

$$\begin{aligned} A &\to E : \{n_1, A\}_{K_E} & & \\ & & E &\to B : \{n_1, A\}_{K_B} \\ & & B &\to E : \{n_1, n_2\}_{K_A} \\ E &\to A : \{n_1, n_2\}_{K_A} & & \\ A &\to E : \{n_2\}_{K_E} & & \\ & & E &\to B : \{n_2\}_{K_B} \end{aligned}$$

This is considered as an attack because B thinks he is communicating with A whereas he is actually talking to E. Note that A and B are both honest in

the sense that they follow faithfully the specification of the protocol and their private keys are not compromised.

Lowe proposed to add B's identity to the second message. This variant is usually referred to as the Needham-Shroeder-Lowe (NSL) protocol, and is currently thought to be secure.

2.2 BAN logic

Introduced by Burrows, Abadi and Needham in [5], BAN logic is a predicate logic for analyzing authentication protocols. Its goal is to prove authentication properties about *honest* principals; secrecy properties are not targeted.

To analyze a protocol in this logic, one first needs to "*idealize*" it. The modified messages—which may include BAN formulas—are supposed to encapsulate the "intended" logical meaning of the messages. Even though intruders are not explicitly considered, this approach allows to find flaws—not automatically, though—in some protocols such as the Needham-Schroeder shared key protocol [23] (not the one mentioned above) or the CCIT X.509 protocol [7]. Honesty is not considered explicitly either; *jurisdiction* formulas are used instead—they can be thought as statements of trust. Complete proofs can be mechanically checked.

However, it has limits as we shall see with the Needham-Schroeder public key protocol. Indeed, the idealized version of this protocol—as given in [5]—is the following:

$$
\begin{aligned}
&A \to B : \{n_1\}_{K_B} \\
&B \to A : \left\{ \left\langle A \overset{n_2}{\rightleftharpoons} B \right\rangle_{n_1} \right\}_{K_A} \\
&A \to B : \left\{ \left\langle A \overset{n_1}{\rightleftharpoons} B, B \models A \overset{n_2}{\rightleftharpoons} B \right\rangle_{n_2} \right\}_{K_B}
\end{aligned}
$$

The process of idealization consists mainly in removing all unnecessary (for the logical inference) data, and turning needed raw data into meaningful BAN formulas. Here, in the second message, B claims that n_2 is a secret shared between A and B, and uses n_1 as a proof of his identity. The third message is an acknowledgment that A effectively received n_2. A actually claims in that message that n_1 is a secret shared between A and B, and B *believes* (denoted by $\models$) that n_2 is a secret shared between A and B. n_2 is used as a proof of A's identity. In this framework, principals are supposed to believe the messages they generate, and proofs are carried out in a Hoare-like [19] fashion: if the protocol consists of messages $m_1, \ldots, m_n$, a proof is a sequence of formulas $\varphi_0, \ldots, \varphi_n$, where φ_{n+1} is obtained from φ_n and the message m_{n+1}, φ_0 representing the initial assumptions. For more detailed information, see [5].

Using standard assumptions—especially the freshness of n_1 and n_2—the following formulas are proved:

$$A \models B \models A \overset{n_2}{\rightleftharpoons} B \qquad B \models A \models A \overset{n_1}{\rightleftharpoons} B$$

at the end of a run of the protocol, which is basically an expression of mutual authentication. However, we have seen that this protocol is not correct, so what

is wrong? Actually, in Lowe's attack, the first formula is intuitively false; the right formula should be

$$A \models E \models A \overset{n_2}{\rightleftharpoons} E$$

If we try to analyze Lowe's variant, we realize that both protocols may have exactly the same idealized version. This is one weakness of BAN logic: the idealization process may add too much information to a protocol. Note, however, that if we first had come up with the idealized protocol, and then converted it to a real one using the recommendations in [5], we would probably have got something closer to Lowe's fix.

Besides, BAN logic does not have inference rules involving directly several steps of a protocol. Each message provides new facts, which are then combined in a purely logical fashion, although some multi-step deductions may be encapsulated in the idealization process. BAN logic cannot deal with properties such as outgoing tests of [16].

2.3 Strand spaces

Strand spaces were introduced by Thayer, Herzog and Guttman in [14]. Unlike BAN logic, there is no idealization step: the reasoning is made directly on the messages of the protocol itself. Each role is clearly separated: each principal runs his or her own program (called *strand*), and the interactions between these programs are under study. This approach, as described in [14], does not use any formal logic system—things are proved "by hand". However, this approach is the starting point of many formal systems, so it is worth mentioning here.

As an example, the NSPK protocol is modelized by two strands:

$$\mathsf{Init}[A, B, n_1, n_2] = \langle + \{n_1, A\}_{K_B}\,, \quad - \{n_1, n_2\}_{K_A}\,, \quad + \{n_2\}_{K_B} \rangle$$
$$\mathsf{Resp}[A, B, n_1, n_2] = \langle - \{n_1, A\}_{K_B}\,, \quad + \{n_1, n_2\}_{K_A}\,, \quad - \{n_2\}_{K_B} \rangle$$

Each strand is a sequence of messages prefixed by a + (for an outgoing message) or a − (for an incoming message). These signed messages can be thought as particles: reactions between same particles of opposite signs are considered.

Roles being considered as independent entities reacting with each other, the modelization of an intruder is straightforward: one just needs to add strands for the intruders abilities. With the presence of the Dolev-Yao intruder, the agreement property is proved for the NSL protocol. The authors of [14] notice that the responder's key may be compromised for the responder's point of view, but no key must be compromised for the initiator's agreement. This proof does not hold for the original NSPK protocol, but it is not clear whether the irrelevance could have been used to find an attack.

Even though proofs in this framework can be checked only by humans, this method has the merit of analysing precisely some hypotheses: the remark about keys stated above could not have been found in other frameworks where the integrity of all keys is always assumed.

2.4 Protocol compositional logic (PCL)

Here, we talk about the protocol logic introduced by Durgin, Mitchell and Pavlovic in [13]. The idea is to provide a logic where proofs of various components can be composed in order to get a proof of an elaborate protocol.

Actually, the same manipulations can also be done in BAN logic, but unlike BAN, secrecy and authentication properties together are targeted. The authors borrow strands from [14], and change them in a more realistic way to get what they call *cords*. For example, the strand-based roles from section 2.3 for NSPK protocol become the following cord-based roles:

$$\mathsf{Init} = (X\,Y)\,[(\nu x)\langle\{x, X\}_Y\rangle(u)(u/\{x, v\}_{\overline{X}})\langle\{v\}_Y\rangle]_X$$
$$\mathsf{Resp} = (X)\,[(x)(x/\{z, Y\}_{\overline{X}})(\nu y)\langle\{z, y\}_Y\rangle(w)(w/\{y\}_{\overline{X}})]_X$$

Here, we identify a principal X and his public key (the private key being $\overline{X}$). Each role is a sequence of actions with a list of static parameters (X and Y for the initiator). The part between brackets is the cord, the index is the principal carrying out the actions. An action consists in generating a new datum (νx), sending a message $\langle m\rangle$, receiving a message (u), or pattern matching $(u/\{x, v\}_{\overline{X}})$. Note that no particular form is assumed at receiving points; decomposition (and decryption) of messages is done by pattern-matching instead. The same notation is used for *traces* (with variables instantiated). Compared to strands spaces, this formalization encapsulates better which items are the parameters to the protocol, which ones are generated, and which ones come from the network.

In addition to this process calculus, [13] also introduces a formal logic system based on first-order predicate logic, with some protocol-specific predicates: Sent, Knows, $\mathsf{Decrypts}$ (which represent principals' action or knowledge), Source (which represent the origin of a datum) and Honest (which modelizes the honesty of a principal).

Predicate formulas and roles are linked together in *modal forms*: a role ρ can be appended a formula ϕ to indicate that ϕ is a consequence of the actions of ρ (this is denoted by $\rho\,\phi$). This is similar to Hoare logic [19]. For example, the following axiom:

$$[(m)]_X\,\exists Y.\mathsf{Sent}(Y, m)$$

means that if some principal X receives a message m, then someone (denoted by Y) must have sent that message. They also introduce a "honesty" rule as an axiom, which allows to link different principals. Intruders are not explicitly modelized.

For the revised NSL protocol, it is then possible to prove the following:

$$(B)[(\nu n)(x)(x/\{m, A\}_{\overline{B}})\langle\{m, n, B\}_A\rangle(y)(y/\{n\}_{\overline{B}})]_B$$
$$\mathsf{Honest}(A) \supset (\mathsf{CSent}(A, \{A, m\}_B) \wedge \mathsf{CSent}(A, \{n\}_B)$$

Here, $\mathsf{CSent}(A, \{n\}_B)$ is a shorthand—defined in terms of Knows and Sent—for *A created and sent* $\{n\}_B$. The formula above can be seen as the responder's part of the agreement property. It does not hold for the original protocol.

This logic has been considerably improved (for example, temporal operators are added in [10]) to get what is now called *protocol composition logic*: modal forms have the more general, Hoare-like, form $\psi\,\rho\,\phi$, as in BAN logic. Under conditions, such modal forms (coming from different protocols) can be composed in various ways, hence the name. Extensions to this framework have been applied to the analysis of IEEE 802.11i and TLS in [18]. There even is a computational semantics for a variant of PCL in [11].

A complete proof in this framework should be mechanically checkable, but no such concrete result has been released so far.

2.5 Basic protocol logic (BPL)

I give an overview of this protocol here for completeness. However, it will be investigated further later. This logic—presented by Hasebe and Okada in [17]—is an extension of first-order predicate logic with equality and subterm relation. Its main purpose is to prove authentication properties. Unlike PCL, whose attempt to prove any kind of property results in a complex logic, basic protocol logic (BPL) remains simple by restricting what can be proved. Another significant difference with PCL is the modelization of honesty: indeed, honesty is no longer an atomic formula, but a compound formula, and there is no "honesty" rule. This simplicity makes the provability of some formulas—called *query forms*—decidable. Moreover, a semantic analysis can reveal practical attacks on flawed protocols.

This theory is recent and still needs improvement. No application to a practical protocol has been released.

2.6 Multiset rewriting (MSR)

This paragraph is not about a specific system, but rather gives an overview of a generic method.

Protocols deal with messages witch can be thought of as resources. Hence, linear-logic-based frameworks may turn out to be useful. This way is explored by Cervesato, Durgin, Lincoln, Mitchell and Scedrov in [8]: the whole system (the network, each principal's memory, etc.) is modelized by a multiset, and each sending action is modelized by a rewriting rule. In the same way multisets and rewriting rules can be seen as linear logic formulas, the generation of a nonce is expressed as an existential quantification on the right hand side; one can use linear logic tools to analyze a protocol formalized in this way. For example, the NSPK protocol would become in terms of linear logic formulas:

$$\begin{aligned}
A_0() &\multimap \exists n_1.N(\{n_1, A\}_{K_B}) \otimes A_1(B, n_1) \\
B_0() \otimes N(\{n_1, A\}_{K_B}) &\multimap \exists n_2.N(\{n_1, n_2\}_{K_A}) \otimes B_1(A, n_1, n_2) \\
A_1(B, n_1) \otimes N(\{n_1, n_2\}_{K_A}) &\multimap N(\{n_2\}_{K_B}) \otimes A_2(B, n_1, n_2) \\
B_1(A, n_1, n_2) \otimes N(\{n_2\}_{K_B}) &\multimap B_2(A, n_1, n_2)
\end{aligned}$$

Here, the predicates A_i, B_i and N represent the states of A, B and the network, respectively. With these formulas, one usually proves that unwanted states are not (or are) reached. This idea has been exploited in an analysis of Kerberos 5 using Isabelle in [6].

3 A word about the computational approach

Even in what is called here the *formal* approach—where encryption is considered as an abstract operation—there are other approaches which do not use proof systems (but, for example, tree automata).

The formal approach presents many advantages such as simple and elegant proofs, the possibility to use existing logic-related tools and sometimes automation. However, the assumptions of the formal approach may seem too naive in real life (consider type flaw attacks). Another research area focuses on the

cryptographic assumptions and their connection to computation. For example, with this *computational* approach, encryption and decryption are no longer considered as symbolic operations, but rather as probabilistic algorithms; an intruder can be any polynomial-time probabilistic algorithm; security properties are expressed in terms of probabilities. The computational approach has been investigated by many researchers since Goldwasser and Micali's work in [15], for example by Bellare, Rogaway et al. in [4] and [3].

Many researchers have worked to link both approaches, such as Lincoln, Mitchell, Mitchell and Scedrov [20], and Abadi and Rogaway [2]. In [2], formal indistinguishability is related to computational indistiguishability: if two expressions are equivalent from the point of view of a formal Dolev-Yao adversary (who cannot guess keys, and can decrypt only with the keys he learns from the expression itself), then a computational adversary can distinguish them only with negligible probability, provided the encryption scheme satisfies some properties. Later, Abadi and Jürjens [1] extend this result to a class of programs which includes most protocols and consider explicitly passive adversaries in protocols (eavesdropping). More recently, Micciancio and Warinschi [22] consider the case of active adversaries.

4 Basic Protocol Logic (BPL)

In this section, I present briefly this logic, and a semantics for which soundness is proved.

4.1 Language

Sorts and terms The language of BPL is order-sorted, and consists of sorts name, nonce and message. $A, B, \ldots, A_1, A_2, \ldots$ ($P, Q, \ldots, P_1, P_2, \ldots$, resp.) are constants (variables, resp.) of sort name, which represent principal names, and $N, N', \ldots, N_1, N_2, \ldots$ ($n, n', \ldots, n_1, n_2, \ldots$, resp.) are constants (variables, resp.) of sort nonce. All terms of sort name and nonce are terms of sort message. The symbols $m, m', \ldots, m_1, m_2, \ldots$ are used to denote variables of sort message. Composed terms of sort message are made by tuples ($\langle m_1, \ldots, m_n \rangle$) and encryption[2] ($\{m\}_P$ or $\{m\}_{P^{-1}}$). We also use the meta-symbols $s, s', \ldots, t, t', \ldots$ to denote any term of sort message. We call the set of terms the *abstract algebra*. Note that all constants are of sort nonce or name.

Formulas We use the structural predicates $s = s'$ (equality) and $s \sqsubseteq s'$ (subterm relation) and three additional *action predicates*: P generates n, P receives m and P sends m. Several action predicates can be combined sequentially to make a *trace formula*. We will denote by $\overrightarrow{\alpha} \equiv \alpha_1, \ldots, \alpha_k$ a trace formula (where each α_i is an action predicate, and k indicates the *length* of $\overrightarrow{\alpha}$). For $\overrightarrow{\alpha} \equiv \alpha_1, \ldots, \alpha_k$ and $\overrightarrow{\beta} \equiv \beta_1, \ldots, \beta_l$, we say that $\overrightarrow{\beta}$ *includes* $\overrightarrow{\alpha}$ (denoted by $\overrightarrow{\alpha} \subseteq \overrightarrow{\beta}$) if there exists a sequence $i_1, \ldots, i_k$ satisfying $0 < i_1 < \cdots < i_k \leqslant l$ and $\overrightarrow{\alpha} \equiv \beta_{i_1}, \ldots, \beta_{i_k}$.

[2] We consider only public key cryptography, but this formalization can be extended to symmetric cryptography as well.

The formulas (denoted by $\varphi, \psi, \dots$) are made by the following grammar:

$$\varphi ::= \overrightarrow{\alpha} \mid m = m' \mid m \sqsubseteq m' \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi \mid \forall x.\varphi \mid \exists x.\varphi$$

Order-preserving merges An *order-preserving merge* of $\overrightarrow{\alpha} \equiv \alpha_1, \dots, \alpha_l$ and $\overrightarrow{\beta} \equiv \beta_1, \dots, \beta_m$ is a trace formula $\overrightarrow{\delta} \equiv \delta_1, \dots, \delta_n$ made by the following rules:

1. $\delta_1 \equiv \alpha_1$ or β_1;
2. for each i $(1 \leqslant i < n)$, if $\alpha_1, \dots, \alpha_j \subseteq \delta_1, \dots, \delta_i$ $(1 \leqslant j < l)$ and $\beta_1, \dots, \beta_k \subseteq \delta_1, \dots, \delta_i$ $(1 \leqslant k < m)$, then $\delta_{i+1} \equiv \alpha_{j+1}$ or β_{k+1};
3. if $\overrightarrow{\delta} \equiv \delta_1, \dots, \delta_i, \delta_{i+1} \dots, \delta_n$ is an order-preserving merge of $\overrightarrow{\alpha}$ and $\overrightarrow{\beta}$ with $\delta_i \equiv \delta_{i+1}$, then $\overrightarrow{\delta'} \equiv \delta_1, \dots, \delta_{i-1}, \delta_{i+1} \dots, \delta_n$ is an order-preserving merge of $\overrightarrow{\alpha}$ and $\overrightarrow{\beta}$.

Query forms [17] introduces *query forms*:

$$Honest(\overrightarrow{\alpha}^P) \wedge \overrightarrow{\beta}^Q \wedge Only(\overrightarrow{\beta}^Q) \rightarrow \overrightarrow{\gamma}$$

where $\overrightarrow{\alpha}^P$ ($\overrightarrow{\beta}^Q$, resp.) is a trace formula representing the actions of the role of a principal P (Q, resp.), $Honest(\overrightarrow{\alpha}^P)$ is a formula expressing the honesty of P—note that in PCL, this is an atomic formula and additional rules are provided—and $Only(\overrightarrow{\beta}^Q)$ a formula expressing that Q performs only the actions of $\overrightarrow{\beta}^Q$. The agreement property (for a bounded number of sessions) can be proved with query forms, and the provabitity of query forms is decidable—this is the main result of [17]. In the following, we do no longer consider query forms.

4.2 Logic

We extend the usual first-order predicate logic with equality by adding the following axioms. This axiomatic system is called *Basic Protocol Logic*.

(I) Axioms of universal sentences over terms When a finite set of literals $\{t_1 = t'_1, \dots, t_n = t'_n, s_1 \sqsubseteq s'_1, \dots, s_j \sqsubseteq s'_j, u_1 \neq u'_1, \dots, u_k \neq u'_k, v_1 \not\sqsubseteq v'_1, \dots, v_l \not\sqsubseteq v'_l\}$ is unsatisfiable in the abstract algebra (with the usual interpretation for $=$ and $\sqsubseteq$), then

$$\forall \overrightarrow{x}.\neg(t_1 = t'_1 \wedge \dots \wedge t_n = t'_n \wedge s_1 \sqsubseteq s'_1 \wedge \dots \wedge s_j \sqsubseteq s'_j \wedge u_1 \neq u'_1 \wedge \dots \wedge u_k \neq u'_k \wedge v_1 \not\sqsubseteq v'_1 \wedge \dots \wedge v_l \not\sqsubseteq v'_l)$$

is an axiom, where $\overrightarrow{x}$ is the set of variables occurring in the litterals. Note that the satisfaction problem is decidable in the free term algebra, hence the set of axioms of type I is recursive.

(II) Rules for trace formulas We use the following axioms[3] for trace formulas:

$\overrightarrow{\beta} \rightarrow \overrightarrow{\alpha}$	when $\overrightarrow{\alpha} \subseteq \overrightarrow{\beta}$
$\overrightarrow{\alpha} \wedge \overrightarrow{\beta} \rightarrow \overrightarrow{\gamma_1} \vee \cdots \vee \overrightarrow{\gamma_m}$	when $\overrightarrow{\gamma_1}, \ldots, \overrightarrow{\gamma_m}$ is the list of order-preserving merges of $\overrightarrow{\alpha}$ and $\overrightarrow{\beta}$

(III) Non-logical axioms [17] also introduces several non-logical axioms expressing properties about protocols. Actually, proving a query form is proving a sequent

$$Honest(\overrightarrow{\alpha}^P), \overrightarrow{\beta}^Q, Only(\overrightarrow{\beta}^Q), \Gamma \vdash \overrightarrow{\gamma}$$

where Γ is the set of non-logical axioms. This set of axioms is important when proving the completeness and the decidability results of [17], but is not important here.

4.3 State-based semantics

In [17], a semantics for BPL is given, and completeness for query forms is proved. This semantics introduces the notion of *trace model*, a tuple $(D_P, D_N, \overrightarrow{\alpha}, \Phi)$, where D_P, D_N are domains for principals and nonces, Φ is an assignment of constants and variables, and $\overrightarrow{\alpha}$ is a sequence of actions (called *trace*) which represents the actions actually performed by the principals).

Here, I reformulate this semantics with sequences of states (called *trajectories*) rather than traces and I prove soundness for this semantics. The semantics is essentially the same, but the different approach is closer to computational analysis of protocol as described in [2].

Definition 4.3.1 (Structure, interpretation)**.** A *structure* is a tuple $(D_P, D_N, \overrightarrow{\sigma}, \Phi)$, where:

- D_P and D_N are two non-empty sets. D_P is called the *principal domain*, and D_N the *nonce domain*. We denote by $\overline{D_P}$ the set $D_P \cup \{\mathsf{Net}_P : P \in D_P\}$, where the Net_P are new symbols, not in D_P. We call *state* a mapping from elements of $\overline{D_P}$ to multisets of ground terms of the multisorted free term algebra over D_P and D_N (which we call the *concrete algebra*).
- Φ maps principal constants to elements of D_P, and nonce constants to elements of D_N.
- $\overrightarrow{\sigma}$ is a sequence of states (called *trajectory*) $s_0, \ldots, s_n$, such that for all $i < n$, there is a unique $P \in \overline{D_P}$ and a m such that $s_{i+1}(P) - s_i(P) = \{m\}$, and one of the following holds:
  - for all $Q \in \overline{D_P} \setminus \{P\}$, $s_{i+1}(Q) = s_i(Q)$
  - $P \in D_P$ and there is a unique $R \in D_P$ such that $s_{i+1}(\mathsf{Net}_R) = s_i(\mathsf{Net}_R) - \{m\}$, and for all $Q \in \overline{D_P} \setminus \{P, \mathsf{Net}_R\}$, $s_{i+1}(Q) = s_i(Q)$

[3] In the original paper, the second axiom was an equivalence, but soundness holds only for one direction. The completeness result actually uses only the "right" direction.

$\mathfrak{S}, \Phi_0 \models_i P$ sends m	iff $s_{i+1}(\mathsf{Net}_P) - s_i(\mathsf{Net}_P) = \{\Phi_0(m)\}$
$\mathfrak{S}, \Phi_0 \models_i P$ generates m	iff $s_{i+1}(P) - s_i(P) = \{\Phi_0(m)\}$ and $s_{i+1}(\mathsf{Net}_Q) = s_i(\mathsf{Net}_Q)$ for all $Q \in D_P$
$\mathfrak{S}, \Phi_0 \models_i P$ receives m	iff $s_{i+1}(P) - s_i(P) = \{\Phi_0(m)\}$ and $s_{i+1}(\mathsf{Net}_Q) = s_i(\mathsf{Net}_Q) - \{\Phi_0(m)\}$ for some $Q \in D_P$
$\mathfrak{S}, \Phi_0 \models \alpha_1; \ldots; \alpha_k$	if there exists a sequence $0 \leqslant p_1 < \cdots < p_k < n$ such that, for all p_i, $\mathfrak{S}, \Phi_0 \models_{p_i} \alpha_i$ holds
$\mathfrak{S}, \Phi_0 \models m = m'$	iff $\Phi_0(m) = \Phi_0(m')$
$\mathfrak{S}, \Phi_0 \models m \sqsubseteq m'$	iff $\Phi_0(m) \sqsubseteq \Phi_0(m')$
$\mathfrak{S}, \Phi_0 \models \neg\varphi$	iff $\mathfrak{S}, \Phi_0 \models \varphi$ does not hold
$\mathfrak{S}, \Phi_0 \models \varphi_1 \wedge \varphi_2$	iff $\mathfrak{S}, \Phi_0 \models \varphi_1$ and $\mathfrak{S}, \Phi_0 \models \varphi_2$
$\mathfrak{S}, \Phi_0 \models \varphi_1 \vee \varphi_2$	iff $\mathfrak{S}, \Phi_0 \models \varphi_1$ or $\mathfrak{S}, \Phi_0 \models \varphi_2$
$\mathfrak{S}, \Phi_0 \models \varphi_1 \rightarrow \varphi_2$	iff $\mathfrak{S}, \Phi_0 \models \varphi_1$ implies $\mathfrak{S}, \Phi_0 \models \varphi_2$
$\mathfrak{S}, \Phi_0 \models \exists x.\varphi$	iff there exists Φ_1 such that Φ_0 and Φ_1 agree on all variables except x and $\mathfrak{S}, \Phi_1 \models \varphi$
$\mathfrak{S}, \Phi_0 \models \forall x.\varphi$	iff for all Φ_1 such that Φ_0 and Φ_1 agree on all variables except x, $\mathfrak{S}, \Phi_1 \models \varphi$

Table 1: State-based semantics for BPL

An *interpretation* is a couple $(\mathfrak{S}, \Phi_0)$, where $\mathfrak{S} = (D_P, D_N, \overrightarrow{\sigma}, \Phi)$ is a structure and Φ_0 maps each variable to D_P or D_N, depending on its sort. Using Φ, we extend Φ_0 to a morphism from the abstract algebra to the concrete algebra.

Henceforth, when we talk about an interpretation $(\mathfrak{S}, \Phi_0)$, we will implicitly use the notations of definition 4.3.1. Intuitively, a state represents the knowledge of the network and the principals: if $P \in D_P$, $m \in P$ means that P either generated or received m, and $m \in \mathsf{Net}_P$ means that P sent m over the network, and nobody received it. More formally:

Definition 4.3.2 (Satisfaction)**.** Let $(\mathfrak{S}, \Phi_0)$ be an interpretation. In table 1, we define the relation $\mathfrak{S}, \Phi_0 \models \varphi$, which is read $(\mathfrak{S}, \Phi_0)$ *satisfies* φ. We say that $\mathfrak{S}$ *satisfies* φ, and we write $\mathfrak{S} \models \varphi$, if $\mathfrak{S}, \Phi_0 \models \varphi$ for all Φ_0. Finally, we say that a structure $\mathfrak{M}$ is a *model* for BPL if it satisfies all non-logical axioms of BPL.

Here is the soundness theorem for this semantics:

Theorem 4.3.3. *If $\Gamma \vdash \Delta$ is provable, then any interpretation which satisfies all formulas of Γ also satisfies one formula of Δ.*

Proof. We prove this by a standard induction on the proof of $\Gamma \vdash \Delta$. Consider each possible last rule:

- $\vdash \overrightarrow{\beta} \rightarrow \overrightarrow{\alpha}$, with $\overrightarrow{\alpha} \subseteq \overrightarrow{\beta}$. Suppose $\mathfrak{S}, \Phi_0 \models \overrightarrow{\beta}$. If $\overrightarrow{\beta} \equiv \beta_1, \ldots, \beta_k$, then there exists a sequence $0 \leqslant p_1 < \cdots \leq p_k < n$ such that, for all p_i, $\mathfrak{S}, \Phi_0 \models_{p_i} \beta_i$ holds. Moreover, since $\overrightarrow{\alpha} \subseteq \overrightarrow{\beta}$, there also exists a sequence

$1 \leqslant q_1 < \cdots < q_l \leqslant k$ such that $\overrightarrow{\alpha} \equiv \beta_{q_1}, \ldots, \beta_{q_l}$ (here, l is the length of $\overrightarrow{\alpha}$). Then the sequence $p_{q_1}, \ldots, p_{q_l}$ meets the requirements for $\mathfrak{S}, \Phi_0 \models \overrightarrow{\alpha}$ to hold. Therefore, $\mathfrak{S}, \Phi_0 \models \overrightarrow{\beta} \rightarrow \overrightarrow{\alpha}$.

- $\vdash \overrightarrow{\alpha} \wedge \overrightarrow{\beta} \rightarrow \overrightarrow{\gamma_1} \vee \cdots \vee \overrightarrow{\gamma_m}$, where $\overrightarrow{\gamma_1}, \ldots, \overrightarrow{\gamma_m}$ is the list of order-preserving merges of $\overrightarrow{\alpha}$ and $\overrightarrow{\beta}$. Suppose $\mathfrak{S}, \Phi_0 \models \overrightarrow{\alpha} \wedge \overrightarrow{\beta}$, with $\overrightarrow{\alpha} = \alpha_1, \ldots, \alpha_k$ and $\overrightarrow{\beta} = \beta_1, \ldots, \beta_l$. Then there exists sequences $0 \leqslant p_1 < \cdots < p_k < n$ and $0 \leqslant q_1 < \cdots < q_l < n$ such that $\mathfrak{S}, \Phi_0 \models_{p_i} \alpha_i$ for all p_i, and $\mathfrak{S}, \Phi_0 \models_{q_j} \beta_j$ for all q_j. Let $r_1 \leqslant r_2 \leqslant \cdots \leqslant r_{k+l}$ be a sorted version of $p_1, \ldots, p_k, q_1, \ldots, q_l$. Let $\overrightarrow{\delta}$ be the action sequence defined by $\delta_i = \alpha_j$ if $r_i = p_j$, or $\delta_i = \beta_j$ if $r_i = q_j$.

  It can be proved that $\overrightarrow{\delta}$ is an order preserving merge of $\overrightarrow{\alpha}$ and $\overrightarrow{\beta}$. Moreover, (consecutive) duplicates in $r_1, \ldots, r_{k+l}$ correspond to consecutive duplicates in $\overrightarrow{\delta}$. Indeed, the restrictions required in definition 4.3.1 are such that, given i, there is a single possible action γ such that $\mathfrak{S}, \Phi_0 \models_i \gamma$. Therefore, when we remove duplicates from $r_1, \ldots, r_{k+l}$ and the matching actions from $\overrightarrow{\delta}$, we get another order-preserving merge of $\overrightarrow{\alpha}$ and $\overrightarrow{\beta}$ — call it $\overrightarrow{\delta'}$ — and a sequence satisfying the conditions for $\mathfrak{S}, \Phi_0 \models \overrightarrow{\delta'}$ to hold. Therefore, $\mathfrak{S}, \Phi_0 \models \overrightarrow{\gamma_1} \vee \cdots \vee \overrightarrow{\gamma_m}$.

The remaining cases are not specific to BPL and can be proved in the general setting of first-order predicate logic with equality (see [24]). □

5 Summary and possible extensions

In this report, several frameworks for verification of cryptographic protocols based on proof systems are presented:

- BAN, historically the first of them, which is quite intuitive and elegant. Although limited, it can highlight interesting facts about protocols;
- strand spaces, less formal than BPL, but much more precise. Even though the proof system itself is not formal, strand spaces introduce formal notations and reasonings for protocols and are the starting point of other formal proofs systems;
- PCL, the most advanced, and also the most intricate one. The whole system is unpractical outside of a proof assistant. An implementation of this theory in Isabelle has been initiated, but it is not really operational;
- BPL, similar to PCL, but considerably simplified. It is also unpractical to use it "on the paper" (maybe even more than PCL, this is a drawback of the more rigourous formalization), and the implementation of a tool (in ML) has been initiated, but is not really operational either. Considering only authentication properties also makes it unable to deal directly with protocols involving session keys (and most of today's protocols involve them). A workaround to this limitation can be additional non-logical axioms.

During this internship, I worked mainly on PCL and BPL. I looked on both implementations. I also considered the problem of bridging these formal approaches and the computational approach in a way similar to Micciancio and Warinschi [22]. Further work may result in the release of a practical proof assistant for BPL and computational results about BPL.

Acknowledgments I thank Mitsuhiro Okada and Koji Hasebe for having welcomed me and introduced me to a new research area, Gergely Bana for the discussions about the computational approach, and all members of Prof. Okada's laboratory in Keio University for their support during my stay in Japan.

References

[1] Martín Abadi and Jan Jürjens. Formal eavesdropping and its computational interpretation. In Naoki Kobayashi and Benjamin C. Pierce, editors, *TACS*, volume 2215 of *Lecture Notes in Computer Science*, pages 82–94. Springer, 2001.

[2] Martín Abadi and Phillip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). In Jan van Leeuwen, Osamu Watanabe, Masami Hagiya, Peter D. Mosses, and Takayasu Ito, editors, *IFIP TCS*, volume 1872 of *Lecture Notes in Computer Science*, pages 3–22. Springer, 2000.

[3] Mihir Bellare, Anand Desai, E. Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. In *FOCS*, pages 394–403, 1997.

[4] Mihir Bellare, Joe Kilian, and Phillip Rogaway. The security of cipher block chaining. In *CRYPTO*, pages 341–358, 1994.

[5] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. In *SOSP '89: Proceedings of the twelfth ACM symposium on Operating systems principles*, pages 1–13, New York, NY, USA, 1989. ACM Press.

[6] Frederic Butler, Iliano Cervesato, Aaron D. Jaggard, and Andre Scedrov. A formal analysis of some properties of Kerberos 5 using MSR. In *Fifteenth Computer Security Foundations Workshop — CSFW-15*, pages 175–190, Cape Breton, NS, Canada, 2002. IEEE Computer Society Press.

[7] CCIT. The directory-authentication framework. *Recommendation X.509*, 1987.

[8] I. Cervesato, N. A. Durgin, P. D. Lincoln, J. C. Mitchell, and A. Scedrov. A meta-notation for protocol analysis. In *CSFW '99: Proceedings of the 1999 IEEE Computer Security Foundations Workshop*, page 55, Washington, DC, USA, 1999. IEEE Computer Society.

[9] John A. Clark and Jeremy L. Jacob. A survey of authentication protocol literature. Technical Report 1.0, 1997.

[10] Anupam Datta, Ante Derek, John C. Mitchell, and Dusko Pavlovic. A derivation system and compositional logic for security protocols. *Journal of Computer Security*, 13(3):423–482, 2005.

[11] Anupam Datta, Ante Derek, John C. Mitchell, Vitaly Shmatikov, and Mathieu Turuani. Probabilistic polynomial-time semantics for a protocol security logic. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP*, volume 3580 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2005.

[12] Danny Dolev and Andrew Chi-Chih Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–207, 1983.

[13] Nancy Durgin, John Mitchell, and Dusko Pavlovic. A compositional logic for proving security properties of protocols. *J. Comput. Secur.*, 11(4):677–721, 2004.

[14] F. Javier Thayer Fábrega, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Why is a security protocol correct? In *Proceedings, 1998 IEEE Symposium on Security and Privacy*, 1998.

[15] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[16] Joshua D. Guttman and F. Javier Thayer Fábrega. Authentication tests. In *SP '00: Proceedings of the 2000 IEEE Symposium on Security and Privacy*, page 96, Washington, DC, USA, 2000. IEEE Computer Society.

[17] Koji Hasebe and Mitsuhiro Okada. Completeness and counter-example generations of a basic protocol logic: (extended abstract). *Electr. Notes Theor. Comput. Sci.*, 147(1):73–92, 2006.

[18] Changhua He, Mukund Sundararajan, Anupam Datta, Ante Derek, and John C. Mitchell. A modular correctness proof of IEEE 802.11i and TLS. In Vijay Atluri, Catherine Meadows, and Ari Juels, editors, *ACM Conference on Computer and Communications Security*, pages 2–15. ACM, 2005.

[19] C. A. R. Hoare. An axiomatic basis for computer programming. *Commun. ACM*, 12(10):576–580, 1969.

[20] Patrick Lincoln, John C. Mitchell, Mark Mitchell, and Andre Scedrov. A probabilistic poly-time framework for protocol analysis. In *ACM Conference on Computer and Communications Security*, pages 112–121, 1998.

[21] Gavin Lowe. An attack on the needham-schroeder public-key authentication protocol. *Inf. Process. Lett.*, 56(3):131–133, 1995.

[22] Daniele Micciancio and Bogdan Warinschi. Soundness of formal encryption in the presence of active adversaries. In Moni Naor, editor, *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 133–151. Springer, 2004.

[23] Roger M. Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers. *Commun. ACM*, 21(12):993–999, 1978.

[24] Gaisi Takeuti. *Proof Theory*, chapter First order predicate calculus, pages 5–74. North-Holland, 1985.