Computer Communications 22 (1999) 697–709

Security protocols over open networks and distributed systems: formal methods for their analysis, design, and verification

S. Gritzalis[a, b,*], D. Spinellis[c], P. Georgiadis[d]

[a]Department of Information and Communication Systems, University of Aegean, Research Unit, 30 Voulgaroktonou St., Athens GR-11472, Greece
[b]Department of Informatics, Technological Educational Institute (T.E.I.) of Athens, Ag. Spiridonos St., Aegaleo, Athens GR-12243, Greece
[c]Department of Information and Communication Systems, University of the Aegean, Samos GR-83200, Greece
[d]Department of Informatics, University of Athens, TYPA Buildings, Athens GR-15771, Greece

Received 22 September 1997; received in revised form 20 November 1998; accepted 20 November 1998

Abstract

Formal methods, theory, and supporting tools can aid the design, analysis, and verification of the security-related and cryptographic protocols used over open networks and distributed systems. The most commonly followed techniques for the application of formal methods for the *ex-post* analysis and verification of cryptographic protocols, as the *analysis approach*, are reviewed, followed by the examination of robustness principles and application limitations. Modern high-level specification languages and tools can be used for automatically analysing cryptographic protocols. Recent research work focuses on the *ex-ante* use of formal methods in the design state of new security protocols, as the *synthesis approach*. Finally, an outline is presented on current trends for the utilisation of formal methods for the analysis and verification of modern complicated protocols and protocol suites for the real commercial world. © 1999 Elsevier Science B.V. All rights reserved.

Keywords: Protocol analysis tools; Formal methods; Security protocols; Cryptographic protocols

1. Introduction

A protocol is a set of rules and conventions that define the communication framework between two or more agents. These agents, known as principals, can be end-users, processes or computing systems. In cryptographic protocols, part of at least one message is encrypted. Security-related and cryptographic protocols are used to establish secure communication over insecure open networks and distributed systems. These protocols use cryptographic techniques to achieve goals such as confidentiality, authentication of principals and services, message integrity, non-repudiation, order and timeliness of the messages, and distribution of cryptographic keys. Unfortunately, open networks and distributed systems are vulnerable to hostile intruders who may try to subvert the protocol design goals.

Given such requirements, it is not surprising that there have been several examples of security-related and cryptographic protocols that were published, believed to be sound, and later shown to have several security flaws [1–3]. After the discovery of flaws in a protocol, the flaws are often corrected or approaches adopted to avoid using the reasoning of the flawed protocols [4]. These facts increasingly prompted research into the development of several different formal methods for detecting protocol failures, following an analysis approach to designing secure protocols. As is the case in the analysis of conventional communication protocols, two kinds of techniques were applied to this problem: those based on attempts to construct inferences using specialised logics based on a notion of knowledge and belief, that protocol participants can confidently reach desired conclusions and, those based on attempts to construct possible attacks using algebraic properties of the algorithms in the protocols.

Inference-construction methods are utilising model logics similar to those that have been developed for the analysis of the evolution of knowledge and belief in distributed systems. These methods are widely used [5–7]. A number of specific problems are associated with them [8–12] related to: the analysis of zero knowledge protocols, the detection of parallel session multi-role flaws, the transformation of messages and prepositions to idealised messages, the fact that there is no complete semantics for the logic, and the modelling of freshness.

* Corresponding author. Tel.: +30-16492-112; fax: +30-16492-299.
E-mail addresses: sgritz@aegean.gr, sgritz@acm.org (S. Gritzalis), dspin@aegean.gr (D. Spinellis), georgiad@di.uoa.gr (P. Georgiadis).

Attack-construction methods construct probable attack sets based on the algebraic properties of the protocol's algorithms. These methods [13–22] are targeted towards ensuring authentication, correctness, or security properties; they are independent of the correctness of a proposed logic. Their main disadvantage lies in the big number of possible events that must be examined.

Attempting to avoid the exponential searches of the attack-construction methods or to extend analyses to protocols that involve arbitrarily large numbers of participants and messages, has given rise to a third approach for the analysis of protocol failures. This is the recent approach of *proof-construction methods*, which has the potential of being as thorough as attack-construction in finding possible attacks, while avoiding exponential searches by replacing them with theorems about these searches. The proof-construction methods are complementary to inference-construction methods, as they are also based on the problem formalisation through hypotheses and authentication properties, but rely on problem-specific properties and a specification at a finer level of precision. Proof-construction methods formally model the actual computations performed in protocols and prove theorems about these computations. This approach has been taken by Snekkenes [23], Bolignano and Paulson [24–26], and Brackin [27].

In this article, we provide a review of the state-of-the-art in the application of formal methods and the development of relevant tools for the analysis, design and verification of security-related and cryptographic protocols and outline major trends of research in this area. The remainder of this article is organised as follows: in Sections 2–4, we describe under the *analysis approach* the most commonly followed approaches to the *ex-post* application of formal methods to the already designed cryptographic protocols. In Section 5, we outline the use of security-related formal specification languages and tools for automatically analysing cryptographic protocols. In Section 6, we present an assortment of helpful principles and limitations encapsulating relative experience of good and bad practice that can be used in the design of error-free cryptographic protocols. In Section 7, we discuss the recent trends for the *ex-ante* use of formal methods in the design stage of new cryptographic protocols using the *synthesis approach*. Finally, in Section 8, we outline the modern trends for the utilisation of formal methods in the analysis and verification of modern complicated protocols and protocol suites for the real commercial world.

2. Attack-construction methods

Attack-construction methods can be divided into three sub-categories based on their theoretical foundation:

- methods based on general purpose validation languages;
- algebraic simplification theoretic model methods; and
- special purpose expert system, scenario based methods.

In the following paragraphs, we present these sub-categories in order to describe the basic features of every method.

2.1. Methods based on general purpose validation languages

These methods analyse a cryptographic protocol as any other program whose correctness they are trying to prove. This is achieved by specifying the protocol: as a finite-state machine [21,22], using predicate calculus [14], or within a process algebra [20,15].

Sidhu and Varadharajan map the protocol to a finite-state machine [21,22]. The first analysis method [21] verifies the basic properties of a number of protocols, detects basic flaws, but is unable to detect flaws due to the re-use of old messages as no temporal assumptions are made. The second method [22] also verifies the basic properties of a number of protocols, but exhibits a number of problems as the number of states increase. In addition, in order to deal with the flaws related to the re-use of old messages, the authors propose to incorporate the analysis data from the session key message contents.

Another approach introduced by Kemmerer, is based on predicate calculus extensions [14]. This method uses the specification language Ina Jo and the Formal Development Methodology. Ina Jo [28] is a non-procedural assertion language that is an extension of first-order predicate calculus. Formal specifications written in Ina Jo specify definitions, initial conditions, transforms, axioms, and criteria. Criteria are used to specify critical requirements for a secure state. Ina Jo formal specifications can then be executed and verified by related tools, such as Inatest. This approach has proved to be successful in locating both active and passive attack flaws, as in both cases the intruder is a separate entity in the model's mathematical framework.

Roscoe proposed a more rigorous approach [20], which is based on modelling all the agents taking part in the protocol, including the communicating principals and the intruder as Communicating Sequential Processes (CSP). The proposed method can be used to formalise messages, traces, intruders, and nonce challenges. The Failures Divergence's Refinement checker (FDR) tool is a general purpose tool that is used later to determine whether an implementation refines a specification. FDR takes as input two CSP processes; a specification and an implementation and tests whether the implementation refines the specification. Initially, this approach was used to analyse many sorts of systems, including distributed databases and communication protocols [29]. Recently, it has also been used to analyse security protocols [20,15]. In the case of protocol authentication, FDR is used to test whether the protocol correctly achieves authentication and to discover a specific kind of attack of the protocol: this is the case in which an intruder is masquerading as another one within a protocol run. Then, the protocol is adapted in order to remove the potential flaw and FDR is used to verify that there are no attacks on a small system running the protocol. The tests using FDR have proven to be rather fast. However, the main problem of the effectiveness of this approach in the examination of large scale systems remains. As FDR requires a user-specified limit on properties such as the number of objects it will consider, failure to find an attack only asserts that an attack cannot be found within those specified limits.

Roscoe and Goldsmith [30] have described how a fully potent cryptographic protocol attacker can be modelled using a given inference system in CSP. Their approach utilises the FDR2 tool by Formal Systems. It uses a lazy exploration strategy which examines the subset of intruder states reachable by the protocol rules effectively exploring the behaviour of the intruder in parallel with the protocol's evolution. A particular advantage of their methodology lies in their ability to reason out the absence of denial-of-service attacks. Their technique requires the production of a CSP description of the protocol by hand. This has proved to be not only time-consuming, but also error-prone, even for experts in this area. For semi-automating the CSP description, Lowe designed a program named Casper [31]. Casper is an effective front-end for the aforementioned approach and will be presented in Section 5.

J.C. Mitchell, M. Mitchell, and Stern use a general-purpose state enumeration tool, named Murϕ [32] (pronounced ''Mur-Phi'' from the Greek letter ϕ) to analyse security-related protocols [19]. The methodology is similar to the approach used in CSP model checking of cryptographic protocols. It involves modelling the protocol and the desired properties in the Murϕ language. Murϕ then verifies—using breadth-first or depth-first full state enumeration—that all reachable states of the system satisfy the specification. Typically, the methodology for analysing protocols involves the following successive steps: formulate the protocol, add an adversary to the system, state the desired correctness condition, run the protocol for some specific choice of system size parameters, experiment with alternate formulation, and repeat. Murϕ has been used to demonstrate flaws already known, as TMN [33] and Kerberos version 5 [34]. A useful aspect of the Murϕ approach is that it is feasible to modify a system description to reflect a situation in which one or more pieces of secret information were compromised.

The standardised language LOTOS [35–36] has also been used to specify security protocols and cryptographic operations [37] and aid the verification of a protocol's robustness to intruder attacks. LOTOS is made up of two main components: a process algebra with a structured operational semantics and an abstract datatype language. The LOTOS formal language has been used to model the Equicrypt protocol [38] for conditional access to multimedia services and to find some successful attacks against it [39]. LOTOS has also been used by Germeau and Leduc to specify a registration protocol for the mutual authentication between a Trusted Third Party and a user [40].

Another general purpose formal specification language used in this area is ASTRAL [41], whose strength lies in the specification real-time systems. Dang uses the ASTRAL model checker [42] to check the ability of satisfaction of critical requirements of an ASTRAL specification by enumerating the possible runs of transitions within a given time. ASTRAL has been applied on the Needham–Schroeder public-key authentication protocol [2], and the TMN protocol [33]. The ASTRAL model checker missed a bug in TMN, because it required excessive execution time under the given ASTRAL coding of the specification. Further, the ASTRAL approach uncovers simple bugs also uncovered by Murϕ tool. These results are preliminary, but it is expected that ASTRAL will prove to be more effective in the investigation of real-time protocols.

All the approaches described before were shown to discover attacks caused by the lack of explicitness in the protocol messages. Unfortunately, they suffer from the large size of the state space under exploration. Additionally, if a method fails to find an attack, this only means that there is no attack on the particular small system analysed, but an attack may exist for some larger system running the same protocol. Hence, the effectiveness of the aforementioned methods in the examination of large scale systems remain to be demonstrated. Lowe [43] presents sufficient conditions for the protocol and its environment guaranteeing that, if there is no breach of secrecy when the protocol is run by an appropriate small system, then there is no breach of secrecy on any system.

Although these general purpose methods were judged as an important contribution to the field, research has often turned into more specialised directions. The driving force behind this turn is the strong desire to use reasoning knowledge specific to the cryptography domain.

2.2. Algebraic simplification theoretic model methods

The algebraic simplification methods model a protocol with a collection of rules for transforming and reducing algebraic expressions representing messages. Representative methods in this category have been proposed by Dolev and Yao [13], and Meadows [16,17].

Dolev and Yao presented the basic model for the state-machine approach [13]. According to their model, an intruder is in full control of the network being able to read, modify, create, and delete messages; effectively, the intruder is using the system being attacked as a machine to generate words (messages). The words follow some rewrite rules, based, for example, on the properties of symmetric encryption. The intruder's task is to discover a word that should have been secret. Thus, the protocol security problem is transformed into a search based on a term-rewrite system. This approach was used to develop analysis algorithms for some restricted protocol classes.

According to the aforementioned work, two models were developed, namely the cascade protocol model, in which the users can apply cryptographic operations in several layers to form messages and, the name-stamp protocol model in which the users are allowed to append, delete, and check names encrypted together with the plaintext. The name-stamp protocol can be used to model layers of encryption. The main drawbacks of the Dolev–Yao model are its failure to model the principals' ability to remember state information between states, and the fact that it can only detect protocol deficiencies.

Meadow's NRL Protocol Analyzer [16,17] is a prototype verification tool, written in Prolog, that can be used to assist either in the verification of security properties of cryptographic protocols or in the detection of security flaws. The NRL model takes the same approach as the term-rewrite model of Dolev–Yao. The main difference between the two models is that the Dolev–Yao model treats a protocol as a machine for producing words, while NRL Protocol Analyzer treats a protocol as a machine for producing not only words, but also beliefs and events. In the NRL model, each protocol participant possesses a set of beliefs. These beliefs are created or modified as a result of receiving messages made up of words, while messages are sent depending on both beliefs and messages received. Events represent the state transitions in which new words are generated and beliefs modified. Thus, an intruder who controls the dissemination of messages can use the protocol to produce words, beliefs, and events.

The NRL Protocol Analyzer, in common with the Interrogator model [18,44] uses a backward search strategy to construct a path from a specified insecure state to an initial state. The main difference between the NRL model and the Interrogator stems from their end goals: the NRL model aims to prove that a protocol is secure, while the Interrogator is designed to search for ways to achieve insecure states without guaranteeing that the protocol is secure if the search fails. However, unlike the Interrogator model the NRL Analyzer can construct a single path using an arbitrary number of protocol rounds, thereby working in an infinite state space. This approach allows the NRL Analyzer to discover attacks based on a combination of a protocol runs.

The NRL Protocol Analyzer was used successfully to locate a series of previously unknown flaws in a number of protocols [45,46], and to demonstrate flaws that were already known in the literature [47]. The main drawback of the current implementation is the fact that to keep the state space workable, some drastic simplifying assumptions are required. In addition, as with most rule-rewrite systems, it is not clear how well the system scales as more complicated algorithms will need to be expressed using an ever increasing set of rules. Another source of difficulty in using the NRL Protocol Analyzer lies in the generation of lemmas stating that infinite classes of states are unreachable: these have to be proved by hand. In Section 5, we describe an effective procedure [48] for making this task easier by automating the process of generation of lemmas.

2.3. Special purpose expert system, scenario based methods

One of the earliest systems that used the Dolev–Yao approach is the Millen's Interrogator Model [18,44]. The Interrogator is a software tool written in Prolog that incorporates a protocol state-transition model. While the abstract model includes the usual state variable for the intruder's set of known items, the search algorithms expressed recursively use a state representation with no explicit mention of the known set.

In addition, the Interrogator has an equation-solving facility for terms using encryption and other operators used in authentication protocols. This facility called generalised narrowing implements a multiple-theory approach which handles commutative operators like exclusive-or and others, such as a limited form of finite-field exponentiation to which prior narrowing algorithms do not apply. Protocol participants are modelled as communicating state machines whose messages to each other are intercepted by an intruder who can either destroy messages, modify them, or let them pass through unmodified. Given a final state in which the intruder knows some word which should be secret, the Interrogator will try—by using operations defined by non-confluent rewrite systems—all possible ways of constructing a path by which that state can be reached. If it finds such a path, then it has identified a security flaw; however, its failure to find an attack does not constitute a proof that no attack exists within its model. The Interrogator model has not uncovered previously unknown attacks in well-known protocols, but it was able to reproduce a number of already known attacks [47].

3. Inference-construction methods

A formal logic model, called BAN logic [5], presented by Burrows, Abadi, and Needham has been widely used in the analysis of authentication protocols. BAN logic of belief belongs to the class of KD45 modal logics which practically means that any fact is only a belief and does not need to be universal in time and space. It assumes that authentication is a function of integrity and freshness, and uses logical rules to trace both of those attributes through the protocol. There are three main stages for the analysis of protocol using the BAN logic. The first step is to express the assumptions and goals as statements in a symbolic notation so that the logic can proceed from a known state to one where it can ascertain whether the goals are in fact reached. The second step is to transform the protocol steps into symbolic notation. Finally, a set of deduction rules called postulates are applied. The postulates should lead from the assumptions, via intermediate formulae, to the authentication goals.

BAN logic was a success. It found flaws in several protocols, including Needham–Schroeder [2] and CCITT X.509 [49]. It uncovered redundancies in many protocols, including Needham–Schroeder, Kerberos [50], Otway–Rees [51], and CCITT X.509 [49]. Many published articles use BAN logic to make claims about their protocol's security [52,53].

Inevitably, critiques on various features of the BAN logic have been published. According to Liebl, it is difficult to prove properties of the BAN logic, such as completeness, and the logic does not take into consideration the release of message contents and the interaction of the runs at different time of the same protocol [54]. Nessett criticised the BAN logic about its claimed goals of authentication [55]. He constructed a specific example in order to demonstrate the BAN logic's failure to discover flaws which violate security in a basic sense. Snekkenes examined the BAN logic's limitation of providing partial correctness of proofs [56]. Syverson described the common misunderstandings about the BAN logic's goals and explained a problem of informality in BAN logic's operational semantics [10]. For this reason, specific measures to formalise BAN logic have been proposed by Mao and Boyd [57]. This formalisation is desirable, not only for its potential in providing rigorous analysis of security protocols, but also for its ability to support computer-aided analysis.

The most criticised points in the BAN logic are: the fact that there is no complete semantics for the logic and the modelling of freshness. The lack of complete semantics may lead to problems in modelling as some facts may have an unclear meaning. It usually causes problems at the idealisation step due to ambiguity and vagueness, particularly where a message is idealised into a formula containing information not present in the message itself. None the less, it was often during the idealisation step that researchers found interesting protocol flaws. An interesting research goal to overcome the BAN logic's drawback would be the development of an efficient method for authenticating protocol idealisations. This method would presumably be based on rule-based techniques and would result in a way to refine a big protocol message transformation step into smaller, simpler, and easier to understand steps. This method would reduce the possibility of error occurrence in the informal protocol idealisation steps and would increase the ease of diagnoses of lower-level design flaws. Mao and Boyd have worked towards this goal [58], but their work neither covers protocols using public-key algorithms nor does it include a theoretic proof of the soundness of the proposed idealisation rules. Regarding the modelling of freshness, it is not possible—as is the case in most modal logics—to distinguish between the freshness of creation and freshness of receipt. The abstract level of BAN logic models results in difficult to assess hypotheses and protocol descriptions. According to Syverson [59], BAN logic's results seem to be less reliable than NRL Protocol Analyzer's, but are easier to come by. Other published logic systems are designed as extensions to BAN logic [6,9,60] or correct perceived weaknesses [56,57].

A successful, but rather complicated approach called GNY logic, was proposed by Gong, Needham, and Yahalom [6] increasing the scope of BAN logic. GNY logic aims to analyse a protocol step-by-step, making explicit any assumptions required, and drawing conclusions about the final position it attains. This logic offers important advantages over the BAN logic. The GNY approach places a strong emphasis on the separation between the content and the meaning of messages which increases the consistency in the analysis and introduces the ability to reason at more than one level. In GNY logic, principals can include messages data which they do not believe in, but just possess. It is also possible to express the ability of a recipient to identify the expected messages and allows one to determine that certain messages are not replays of a recipient's own previous messages in a given session. GNY logic has a number of drawbacks: it addresses only authentication and is much more complicated and elaborate than other methods as it has many rules which have to be considered at each stage [61].

A Higher Order Logic (HOL) theory [62] formalising an extended version of GNY, named BGNY logic has been introduced by Brackin [8]. This belief logic is used by the software that automatically proves authentication properties of cryptographic protocols. Similar to the GNY logic, BGNY addresses only authentication. However, BGNY extends the GNY logic including the ability to specify protocol properties at intermediate stages, and being able to specify the protocols that use multiple encryption and hash operations, message authentication codes, hash codes as keys, and key-exchange algorithms.

Another logic, called SvO, presented by Syverson and van Oorschot [7], is designed to capture the features of extensions and variants of four logics, namely BAN, GNY, AT [63], and vO [60] in a single unified framework. In addition, the authors provide model-theoretic semantics with respect to which the logic is sound. The SvO logic was intended to encompass the reasoning of these other logics, while providing a rigorous understanding of its formal expressions. The SvO logic is considered to be simpler to use and more expressive than any of the logics from which it is derived.

Kailar proposes a special-purpose logic to be used in the analysis of communication protocols that require accountability [64], such as those for secure electronic transactions. This logic looks at what can be achieved without making any assumptions about freshness. A set of postulates which are applicable to the analysis of proofs in general, and the proofs of accountability in particular are proposed. In the same framework, an authentication logic presented by Kessler and Neumann [65] analyse the accountability of transactions in the framework of electronic commerce protocols. Their work is based on the AUTLOG semantics developed in [66]. New rules and predicates are used to model accountability and to prove that the new calculus is correct with respect to the formal semantics.

Wedel and Kessler also propose a logic for the analysis of authentication protocols [66] providing formal semantics for proving its soundness. This logic can handle a wide variety of cryptographic mechanisms using a minimum of notation. In their approach, the elimination of the formuli out of the idealised messages leads to a clear distinction between the protocol itself and the assumptions about it.

4. Proof-construction methods

As mentioned before, inference-construction methods do not address secrecy, often lack clear semantics, and it is sometimes difficult to say exactly what a belief-logic proof actually proves. However, attack-construction methods may have to search spaces that grow exponentially with the size of the protocol, hence the time and space they require can easily exceed all available resources.

In order to confront the aforementioned drawbacks, Bolignano proposed an approach targeting the generation of human-readable proofs [25]. Such proofs can be used as part of a vulnerability analysis or formal code inspections. In order to achieve this goal, specific properties of the problem are used to formalise the requirements and simplify the proofs. The approach places particular emphasis on the clear description of the problem providing a clear separation between reliable and unreliable principals, the precise description of their roles, beliefs, and control structures, the imposition of sequencing constraints, the expression of authentication properties using temporal features, and the formalisation of algorithm properties. The use of powerful invariants and the axiomatisation of intruder knowledge result in a verification process whose conciseness is comparable to that of modal based approaches. The process can be automated within a framework of typed logics using the Coq proof assistant [67].

Paulson has independently developed a similar approach synthesising the inference-construction method idea of protocol message guarantees and the attack-construction method notion of events [24,26]. Unlike Bolignano, who models the states of the four agents A, B, S, and the Spy, Paulson defines protocols inductively as the set of all possible event traces. Agents receiving a trace can forward it and extend it according to the protocol rules, while remaining agnostic to the message's true sender. This approach allows the modelling of both attacks and key losses. Again, the process is partially automated using the Isabelle theorem prover [68].

Within the same framework, Schneider presents a general approach for the analysis and verification of authentication properties in the language of CSP [69]. The focus of this research work is the development of a specific theory targeted towards the analysis of authentication protocols and built on top of the general CSP semantic framework. The CSP syntax provides a precise way to describe authentication protocols in terms of the messages accepted and transmitted by the individual protocol participants. This approach aims to bridge the gap between the ability to express authentication protocols in a precise way and the facility to reason formally about the properties they exhibit. The aforementioned theory has been successfully applied [70] to the modelling and analysis of the Zhou and Gollmann fair non-repudiation protocol [71].

More recently, Fabrega, Herzog, and Guttman introduced the notion of a *strand space* [72,73]. They proposed a model and a set of proof methods for cryptographic protocols along the lines of the NRL Protocol Analyzer, Schneider's work, and Paulson's inductive definitions. A *strand* is defined as a sequence of events within the domain of a security protocol representing the actions of legitimate party or a penetrator. Defining a *strand space* as a graph of stands allows protocol correctness to be expressed as connections between different kinds of strands. In conjunction with the aforementioned formalism, the authors use the concept of ideals to prove the bounds on a penetrator's capabilities independent of the security protocol being analysed. This approach is characterised by the simplicity of the model and the effortless production of reliable protocol correctness proofs. An interesting concept of this method is the pictorial approach [74] which is used as a heuristic for stating and proving the correctness results. Strand spaces can help users draw informative pictures of security protocols, attacks, correctness theorems, and crucial steps in the proofs; thereby focusing on the protocol goals and their satisfaction.

Finally, another related approach is the implementation of the protocol security theory developed by Snekkenes [23] within the HOL theorem prover [62]. The prover works on an explicit identification of the participants' message extractions, computations, tests, and actions exploiting the algebraic properties of the cryptographic algorithms used. The theory also distinguishes participants from the roles they play, thereby modelling attacks in which an attacker leads the legitimate participants to think they are at different stages in the protocol than they really are Snekkenes has also developed proof tools that are based on large amounts of human input.

5. Formal specification languages and tools for automatically analysing cryptographic protocols

The techniques outlined before cannot be easily applied by analysts other than the developers themselves. The main reason for this difficulty is the fact that the protocols have to be respecified for each technique, and it is not easy to transform the published description of the protocol into the required formal system. Hence, some tools are designed as automatic translators. The input to any such translator still requires a formally-defined language, but it can be made similar to the message-oriented protocol descriptions that are typically published in the literature. This introduces the idea of designing a single common protocol specification language that could be used as the input format for any formal analysis technique.

Meadows [48] describes a heuristic procedure for automating language generation using the NRL Protocol Analyzer. In contrast to other methods, in which languages are defined by hand and unreachability properties are automatically proved, this approach combines the language generation with the unreachability proof. By using this approach, the languages are efficiently created and are also amenable to automated analysis thereby improving the performance of the NRL Protocol Analyzer.

Brackin specifies a simple Interface Specification Language (ISL) and describes an Automatic Authentication Protocol Analyzer (AAPA) which can automatically either prove that specific protocols satisfy the desired properties, or identify precisely where these proof attempts fail [75–77]. The AAPA produces its proofs using the BGNY protocol analysis belief logic implemented in the HOL family of proof tools. The AAPA can be used either alone or as part of the Convince system [78]. The Convince tool facilitates the modelling and analysis of cryptographic protocols using a HOL theorem prover with automated support. This tool, developed by Lichota, Hammonds, and Brackin implements BGNY belief logic. It consists of the AAPA together with a graphical user interface that automatically creates ISL specifications from user-created graphical protocol representation.

The time and space required to do an AAPA analysis grow quadratically with the size of the protocol making it possible for the AAPA to quickly analyse large and complicated protocols. A creditable performance to evaluate the results of AAPA includes the analysis of 52 protocols from ‘‘A Survey of Authentication Protocol Literature’’ by Clark and Jacob [79]. This is a continually updated library of protocols analysed in the protocol-failure literature. As mentioned before, the time for protocol analysis proved to be quite brief; an experienced user needed 80 working hours to model and analyse 52 protocols. However, AAPA misses some failures, most notably non-disclosure failures, due to the fact that BGNY belief logic makes authentication deductions by assuming that there were no non-disclosure violations. Nevertheless, AAPA remains one of the most effective modern tools aiding the design and analysis process.

Another promising language named Common Authentication Protocol Specification Language (CAPSL), partly inspired by ISL, is being developed by Millen [80]. CAPSL is proposed as a single common protocol specification language that can be used as the input format for any formal analysis technique, such as Prolog state-search analysis tools [18], the NRL Protocol Analyzer [16,17], model-checking with FDR [15], and HOL [8]. The main objectives of the CAPSL design are usability, abstraction, completeness, extensibility, parsability, and scalability.

A program named Casper, developed by Lowe [31], semi-automatically produces the CSP description from a more abstract description, thus greatly simplifying the modelling and analysis process. Casper compiles high level protocol descriptions, written following a notation similar to the notation used in the academic literature, into CSP scripts for checking on FDR2. Casper does not yet cover all the features found in the security protocols, but it was applied to a number of known protocols, such as the Andrew protocol, the Kerberos protocol, the CCITT X.509 protocol, and the Yahalom protocol [5]. Some of these case studies are available via the Casper Web page [81]. The main differences between the CAPSL and the Casper language stem from the fact that a Casper input file has to define not only the protocol itself, but also the system to be checked; CAPSL defines only the protocol. The most important consequence for Casper is that when designing the input syntax, one has to provide a mechanism for defining the agents who are taking part in the system, the specific roles they played, and the data items they used.

Recently, Abadi and Gordon developed the Spi calculus language to be used for specifying the cryptographic protocols [82]. The Spi calculus is an extension of the Pi calculus [83], an existing language for specifying mobile computation which does not include any construction for encryption and decryption. Within Spi calculus protocols are represented as processes, while their security properties are represented using the notion of protocol equivalence. The Spi calculus approach resembles the modal logic reasoning about channel utterances and the characterisation of knowledge within a state-machine environment. However, it differs from other approaches by representing integrity and secrecy as equivalencies and defining the environment as a Spi calculus process.

Finally, Brackin proposed a tool targeted towards automating proof-construction [27,84]. The HOL theory Protocol Description Logic (PDL) formalises the low-level details of the actions performed by processes executing a protocol. PDL is not sufficiently expressive to formalise all protocols specified in CAPSL [80]. The main advantage of Brackin’s approach is that the complexities that arise in proving desired belief inferences will not need to be considered again once these inferences are proved. Thereby protocol analyses using PDL are not only as trustworthy as those produced by attack-construction methods, but also comparable in speed to those performed using inference-construction methods.

6. Robustness principles

Aiming towards the design of an effective cryptographic protocol, a complementary approach is to try to encapsulate experience of good and bad practice into empirical rules. The robustness principles are therefore helpful, in that adherence to them contributes to the simplicity of protocols and avoids a considerable number of published confusions and mistakes. Anderson and Needham [85] propose a number of robustness principles, and Abadi and Needham [86,87] introduce complete analyses of desirable protocol properties and relevant limitations. Some of them are mentioned in the following:

- be very clear about the security goals and assumptions;
- be clear about the purpose of encryption (secrecy, authenticity, etc.), do not assume that its use is synonymous with security;
- be careful that your protocol does not make some unexamined assumption about the properties of the underlying cryptographic algorithm;
- be sure to distinguish different protocol runs from each other;
- do not assume that a message you receive has only a particular form, even if you can check this;
- if timestamps are used as freshness guarantees by reference to absolute time, then the difference between local clocks at various machines must be less than the allowable age of a message deemed to be valid; further, the time maintenance mechanism everywhere becomes part of the Trusted Computing Base;
- where the identity of a principal is essential to the meaning of a message, it should be mentioned explicitly in the message;
- sign before encrypting; if a signature is affixed to encrypted data, then one cannot assume that the signer has any knowledge of the data; a third party certainly cannot assume that the signature is authentic, so non-repudiation is lost.

It is remarkable that, in many cases, following one design principle will sometimes lead to violating another. This is almost expected, as we have to deal with empirical rules. In addition, even following all these rules will not guarantee a sound design. Many authors have considered the question of what are appropriate goals in the context of protocol analysis. Accordingly, Boyd [88] reviewed some design goals in authentication protocols and proposed a classification of them: intentional and extensional goals. Intentional goals are generally concerned with ensuring that the protocol runs correctly as specified, while extensional goals are concerned with what the protocol achieves for its participants. It was suggested that attacks should be measured by whether or not they violate extensional specifications even if intentional ones were used to find the attacks in the first place. Boyd proposes a hierarchy of extensional protocol goals which includes the major proposed goals for key establishment. He further demonstrates how these extensional goals can be exploited to motivate design of entity authentication protocols.

Concluding, it is becoming widely accepted that both formal methods and structured design rules must be taken into account in a complementary way during all phases of a protocol design for achieving effective and reliable cryptographic protocols.

7. Formal methods for protocol design

The design of secure cryptographic protocols is a very complex and difficult process. Until recently, researchers were orientated towards the use of formal methods for the analysis and verification of existing protocols. These methods have proved successful in discovering flaws with existing protocols, sometimes previously unrecognised ones. Criticisms on formal verification include the fact that key distribution protocols, claimed to be secure under BAN logic, have already been broken [89]. Further, BAN-like logics do not prove that a weakness in the protocol implies that the cryptoscheme, on which it is based, can be broken. Therefore, a great deal of doubt remains as to whether any of the existing techniques is sufficient to provide a proof that a given protocol is correct. This situation has a fair analogy in the verification process of general purpose computer programs, in which reliable testing techniques allows us to find many bugs, but will not provide a basis for complete proof of correctness. Therefore, it would be a prudent and mature trend, to design specific methods and implement tools, in order to aid the initial correct design of cryptographic protocols. The incorporation of formal methods into design can be implemented in various ways.

One approach is to develop and use protocol design methodologies that lend themselves to formal method analysis [90]. This is exemplified by the modular design proposed by Heintze and Tygar [91]. Using protocol security reasoning tools and a composition theorem, they can state sufficient conditions for combining two secure protocols to form a new one with similar properties. Based on secret-security and time-security notions they can provide examples of how unmet conditions result in an insecure protocol.

Another approach is based on the development of design principles which are used to develop protocols whose security is easy to evaluate. To satisfy this goal, Gong and Syverson propose the notion of fail-stop protocols [92]. The main idea was derived from an earlier work where they proposed the concept of a fail-stop processor, which, when failing, stops before any effect is visible to the outside environment [93]. Similarly, a fail-stop protocol halts in response to active attacks interfering with the protocol execution. Given such a protocol, its security analysis involves only the examination of possible passive attacks such as eavesdropping. It is, therefore, much easier to conclude whether the secrecy assumption can be violated. The suggested proof methodology for a fail-stop protocol comprises three phases: the verification that the protocol is fail-stop, the validation of the secrecy assumption, and the application of BAN-like logics. The proposed methodology applies the BAN-like logics because even for a fail-stop protocol, the residue from its execution may be useful to an attacker [55]. Another encouraging point for this methodology is that the specifications of fail-stop protocols satisfy some of the main prudent engineering principles from Refs. [86,87]. Accordingly, the GNY logic is used to analyse a fail-stop protocol the proof complexity can be dramatically reduced. According to the researcher's team investigation, many existing protocols proved to be fail-stop [92]; therefore the new notions are not too limiting.

The most prominent approach in this area seems to be the layered approach proposed by Meadows [90]. This approach can be used together with Heintze and Tygar's approach [91]. It is based on a stack of models at different levels of abstraction. As a first step, the protocol designer uses a relatively abstract model to construct and verify the security protocol. If this protocol is correct at the top layer, the designer focuses on a more detailed model which refines the abstract one. The repeated execution of this process leads to the final production of a detailed specification. Much of the existing work on requirements specifications [11] has this specific flavour. For the application of BAN logic [94], the approach is based on a parser that translates members of a limited class of protocol specifications into BAN logic.

Within this framework, Rudolph introduces an approach for designing an abstract model for cryptographic protocols that can be used as the top layer of a layered design method [95]. The main idea of Rudolph is the usage of Asynchronous Product Automata. The whole design process starts with a relatively abstract model at the top layer and ends in a refined specification that can be proved to be an implementation of the top level. This model reaches a higher level of abstraction than the model presented in the work of Heintze and Tygar [91] through the use of logical secure channels instead of encryption.

The notion of channels is also utilised by Buttyan, Staamann, and Wilhelm to present a simple logic for authentication protocol design [96]. These channels are abstract views of various types of secure communication links between principals. The way channels are used is similar to how the Pi-calculus channel primitives are used in the Spi-calculus paper [82]. The proposed Simple Logic preserves the simplicity of BAN logic and adopts some concepts from GNY logic. It consists of a language and a small number of inference rules. The language is used to describe assumptions, events, and the protocol goals. The inference rules are used to derive new statements about the system. The goal of the analysis is to construct a witnessing deduction, which is a derivation of the goals from the assumptions and the formal protocol description. The protocol is correct in the case where such a deduction exists. The lack of a witnessing deduction means that the protocol may not be correct. The main advantage of this approach is the fact that in this logic encryption and keys are replaced by channels with various access restrictions. This abstraction makes the logic compact because the cryptographic operations that are usually used in authentication protocols can all be described with a uniform notation. Furthermore, channels enable thinking about protocols at a high abstraction level without being concerned with the implementation details. Thus this logic can be used for constructing protocols at the first place rather than verifying existing protocols. For this reason they proposed synthetic rules derived from the logic, that can be used by designers to construct protocols in a systematic way.

Boyd and Mao proposed another technique for designing key exchange protocols [97] which are guaranteed to be correct in the sense that a specified security criterion will not be violated if protocol principals act correctly. This technique is developed from basic cryptographic properties that can be expected to be held by a variety of cryptographic algorithms. Protocols can be developed abstractly and any particular type of algorithm that possesses the required property can then be used in a concrete implementation.

Gollmann [98] suggests that the design of authentication protocols has proven to be error prone partly due to a language problem. The objectives of entity authentication are usually given in terms of human encounters, while we actually implement the message passing protocols. The author proposes various translations of the high level objectives into a language appropriate for communication protocols.

Several researchers believe that in the near future, more effort will be spent on designing secure protocols and less on formal verifications. As expected, this trend has received criticism similar in nature to that expressed towards the use of formal methods in program design and implementation [99]. Specifically, Meadows argues [90] that design specifications do not guarantee that protocols will meet security goals that were not foreseen by the design approach, that the protocols designed are sometimes impractical, and that—due to the imprecision of design principles—flawed protocols may in any case be designed.

8. Conclusions

We presented an overview of the modern trends in the application of formal methods for the analysis and verification of cryptographic protocols. The three method families we described are useful at various levels of abstraction. The more abstract models can be used efficiently at earlier points in the design stage, when implementation details have not yet been decided. A protocol analysis toolkit-based usage scenario can be described as follows [24,25,90]: initially, use an inference-construction method, like BAN, to determine what the role of each message of a protocol should be and to ensure freshness properties; then, use an attack-construction method, like NRL Protocol Analyzer, for finding simple attacks quickly; and finally, utilise a proof-construction method to investigate the deeper properties with a modest amount of effort.

Further, some areas where current research is conducted have emerged. Bolignano is working towards investigating the use of his approach in the context of an ITSEC evaluation. Another interesting research direction is the investigation of the potential integration of methods like the NRL Protocol Analyzer and the Interrogator model within the methodology of fail-stop protocols in the cases of protocols that do not satisfy the fail-stop requirements [100]. To ease and extend the use of the Murϕ tool, it would be useful to achieve automatic translation of a higher-level protocol specification language such as CAPSL into Murϕ and combine analyses using exhaustive finite-state analysis and formal logic methods.

Moreover, the research community is also working towards developing tools that take easy-to-write specifications of protocols and the expected properties and quickly perform formal analyses checking for failures of these protocols to achieve their desired properties. AAPA [75] and CAPSL [80] seem to be the most promising approaches to bridge the gap between the typical informal presentations of protocols given in research articles and the precise characterisations required to conduct formal analysis. Representative research and development attempts for designing effective tools, include the work of Brackin for a new, currently unnamed, protocol analysis tool which, unlike AAPA, will use CAPSL rather than ISL as its input language and use the Protocol Description Language as the basis for its proofs. In addition, the co-operation of Millen, Meadows, and Brackin has resulted in a—yet unpublished—multi-purpose CAPSL translator [84]. This translator will be able to translate CAPSL protocol specifications into a HOL theory to be used with PDL, into input of the NRL Protocol Analyzer, or into human-readable algorithm descriptions.

An interesting research trend, lies in the fact that many current activities use formal methods for analysing and verifying modern protocols and protocol suites to be used in the commercial world. These suites consist of a set of single protocols which interact with each other causing, previously unknown, potential vulnerabilities. Within this context, Brackin describes how the AAPA works to formally analyse two large commercial protocols [101]: the main- and coin-sequence protocols developed by CyberCash, Inc., Murϕ was used in order to analyse the SSL 3.0—a *de facto* standard for achieving secure Internet communication—handshake protocol [102], and Paulson's Inductive method has been applied [103] to analyse the descendant of SSL 3.0, known as TLS Internet Protocol. Recently, Paulson's Inductive method has also been exploited [104,105] to formalise Kerberos version IV, a real-world timestamp-based protocol. In this work, a complete formalisation of the whole protocol is achieved, and several guarantees about its entangled operation are proved using the Isabelle theorem prover. Further, Bolignano generalised [106] his earlier approach [25] in order to achieve the verification of electronic payment protocols, such as C-SET and SET. The NRL Protocol Analyzer has also been utilised by Meadows for the analysis and verification [107] of the Internet Key Exchange protocol, IKE (formerly ISAKMP/Oakley) [108].

Finally, from the security protocol designer's point of view, the research community, working towards developing more effective techniques to *ex-ante* design protocols that are guaranteed to be reliable and correct in the first place, has implemented the synthesis approach. Most of the recent research in this area is focused on the application of the notion of channels in order to effectively implement the layered approach.

Acknowledgements

The authors would like to thank C. Meadows, Naval Research Laboratory, S.H. Brackin, ARCA Systems, Inc., L.C. Paulson, University of Cambridge, C. Boyd, Queensland University of Technology, W. Mao, Hewlett-Packard, and the anonymous referees for their insightful remarks and constructive comments.

References

[1] D. Denning, G. Sacco, Timestamps in key distribution protocols, Communications of the ACM 24 (8) (1981) 533–536.
[2] R. Needham, M. Schroeder, Using encryption for authentication in large networks of computers, Communications of the ACM 21 (12) (1978) 993–999.
[3] R. Needham, M. Schroeder, Authentication revisited, Operating Systems Review 21 (1) (1987) 7.
[4] A. Patel, S. O'Connell, A timestamp model for determining real-time communications in intelligent networks, Computer Communications 20 (1997) 211–218.
[5] M. Burrows, M. Abadi, R. Needham, A logic of authentication, ACM Transactions on Computer Systems 8 (1) (1990) 18–36.
[6] L. Gong, R. Needham, R. Yahalom, Reasoning about belief in cryptographic protocols Proceedings of the 1990 IEEE Symposium on Security and Privacy, IEEE Computer Society, Silver Spring, MD, 1990 pp. 234–248.
[7] P. Syverson, P.C. van Oorschot, On unifying some cryptographic protocol logics Proceedings of the 1994 IEEE Computer Security Foundations Workshop VII, IEEE Computer Society, Silver Spring, MD, 1994 pp. 14–29.
[8] S. Brackin, A HOL extension of GNY for automatically analyzing cryptographic protocols Proceedings of the 1996 IEEE Computer Security Foundations Workshop IX, IEEE Computer Society, Silver Spring, MD, 1996 pp. 62–76.
[9] V. Kessler, W. Wedel, AUTLOG—an advanced logic of authentication Proceedings of the 1994 IEEE Computer Security Foundations Workshop VII, IEEE Computer Society, Silver Spring, MD, 1994 pp. 90–99.
[10] P. Syverson, The use of logic in the analysis of cryptographic protocols Proceedings of the 1991 IEEE Computer Security Symposium on Security and Privacy, IEEE Computer Society, Silver Spring, MD, 1991 pp. 156–170.
[11] P. Syverson, C. Meadows, A logical language for specifying cryptographic protocol requirements Proceedings of the 1993 IEEE Computer Security Symposium on Security and Privacy, IEEE Computer Society, Silver Spring, MD, 1993 pp. 165–177.
[12] S. Gritzalis, The BAN logic for the analysis of authentication protocols in distributed systems: a review, in: Proceedings of the 1st meeting of the IKAROS Greek Computer Society's human network for the Security, Quality, and Reliability in Information and Communication Technologies, 1996 (in Greek).
[13] D. Dolev, A. Yao, On the security of public key protocols, IEEE Transactions on Information Theory 29 (2) (1983) 198–208.
[14] R. Kemmerer, Analyzing encryption protocols using formal verification techniques, IEEE Journal on Selected Areas in Communications 7 (4) (1989) 448–457.
[15] G. Lowe, Breaking and fixing the Needham-Schroeder public-key protocol using FDR Proceedings of TACAS, Springer, Berlin, 1996 pp. 147–166.

[16] C. Meadows, Applying formal methods to the analysis of a key-management protocol, Journal of Computer Security 1 (1992) 5–35.

[17] C. Meadows, The NRL protocol analyzer: an overview, Journal of Logic Programming 26 (2) (1996) 113–131.

[18] J. Millen, The interrogator model Proceedings of the 1995 IEEE Symposium on Security and Privacy, IEEE Computer Society, Silver Spring, MD, 1995 pp. 251–260.

[19] J.C. Mitchell, M. Mitchell, U. Stern, Automated analysis of cryptographic protocols using Murϕ Proceedings of the 1997 IEEE Symposium on Security and Privacy, IEEE Computer Society, Silver Spring, MD, 1997 pp. 141–151.

[20] A.W. Roscoe, Modelling and verifying key-exchange protocols using CSP and FDR Proceedings of the 1995 IEEE Computer Security Foundations Workshop IIX, IEEE Computer Society, Silver Spring, MD, 1995 pp. 98–107.

[21] D. Sidhu, Authentication protocols for computer networks, Computer Networks and ISDN Systems 11 (1986) 297–310.

[22] V. Varadharajan, Verification of network security protocols, Computers and Security 8 (1989) 693–708 Elsevier Advanced Technology.

[23] E. Snekkenes, Formal Specification and Analysis of Cryptographic Protocols, PhD thesis, University of Oslo, Norway, 1995.

[24] L. Paulson, Proving properties of security protocols by induction Proceedings of the IEEE Computer Security Foundations Workshop X, IEEE Computer Society, Silver Spring, MD, 1997 pp. 70–83.

[25] D. Bolignano, An approach to the formal verification of cryptographic protocols Proceedings of the Third ACM Conference on Computer and Communications Security, ACM, New York, 1996 pp. 106–118.

[26] L. Paulson, Mechanized proofs for a recursive authentication protocol Proceedings of the IEEE Computer Security Foundations Workshop X, IEEE Computer Society, Silver Spring, MD, 1997 pp. 84–94.

[27] S. Brackin, A State-Based HOL Theory of Protocol Failure, ATR 98007, Arca Systems, Inc., 1997, http://www.arca.com/paper.htm.

[28] J. Scheid, S. Holtsberg, Ina Jo Specification Language Reference Manual, System Development Group, Unisys Corporation, CA, 1988.

[29] A.W. Roscoe, Developing and verifying protocols in CSP, Proceedings of Mierlo workshop on protocols, TU Eidhoven, 1993.

[30] A. Roscoe, M. Goldsmith, The perfect spy for model-checking cryptoprotocols, in: Proceedings of the 1997 DIMACS Workshop on Design and Formal Verification of Security Protocols, 1997, http://dimacs.rutgers.edu/Workshops/Security/.

[31] G. Lowe, Casper: a compiler for the analysis of security protocols Proceedings of the 1997 IEEE Computer Security Foundations Workshop X, IEEE Computer Society, Silver Spring, MD, 1997 pp. 18–30.

[32] D.L. Dill, A.J. Drexler, A.J. Hu, C.H. Yang, Protocol verification as a hardware design aid Proceedings of the 1992 IEEE International Conference on Computer Design: VLSI in Computers and Processors, IEEE Computer Society, Silver Spring, MD, 1992 pp. 522–525.

[33] M. Tatebayashi, N. Matsuzaki, D. Neuman, Key distribution protocol for digital mobile communication systems Proceedings of CRYPTO ’89, Springer, Berlin, 1990 pp. 324–333.

[34] J. Kohl, B. Neuman, T. Ts’o, The evolution of the Kerberos authentication service ver. 5, in: F. Brazier, D. Johansen (Eds.), Distributed Open Systems, IEEE Computer Society, Silver Spring, MD, 1994 pp. 78–94.

[35] T. Bolognesi, E. Brinksma, Introduction to the ISO specification language LOTOS, Computer Networks and ISDN Systems 14 (1987) 25–59.

[36] ISO/IEC Information Processing Systems—Open Systems Interconnection: LOTOS, a Formal Description Technique based on the Temporal Ordering of Observational Behaviour, IS8807, 1989.

[37] F. Germeau, G. Leduc, Model-based Design and Verification of security protocols using LOTOS, in: Proceedings of the 1997 DIMACS Workshop on Design and Formal Verification of Security Protocols, 1997, http://dimacs.rutgers.edu/Workshops/Security/.

[38] J. Guimaraes, J. Boucqueau, B. Macq, OKAPI: a Kernel for Access Control to Multimedia services based on TTPs, in: Proceedings of the 1996 ECMAST European Conference on Multime-dia Applications, Services and Techniques, 1996, pp. 783–798.

[39] G. Leduc, O. Bonaventure, E. Koerner, L. Leonard, C. Pecheur, D. Zanetti, Specification and Verification of a TTP protocol for the conditional access to services, in: Proceedings of the 12th J. Cartier Workshop on Formal Methods and their applications: Telecommunications, VLSI and Real-Time Computerised Control Systems, 1996.

[40] A computer-aided design pf a secure registration protocol, in: F. Germeau, G. Leduc (Eds.), Formal Description Techniques and Protocol Specifications, Testing and Verification, FORTE/PSTV’97, Chapman and Hall, London, 1997, pp. 145.

[41] C. Ghezzi, R. Kemmerer, ASTRAL: an assertion language for specifying real-time systems, in: Proceedings of the Third European Software Engineering Conference, 1991, pp. 122–146.

[42] Z. Dang, Using the ASTRAL Model Checker for Cryptographic Protocol Analysis, in: Proceedings of the 1997 DIMACS Workshop on Design and Formal Verification of Security Protocols, 1997, http://dimacs.rutgers.edu/Workshops/Security/.

[43] G. Lowe, Towards a completeness result for model checking of security protocols Proceedings of the IEEE Computer Security Foundations Workshop XI, IEEE Computer Society, Silver Spring, MD, 1998 pp. 96–105.

[44] J. Millen, S. Clark, S. Freedman, The interrogator: protocol security analysis, IEEE Transactions on Software Engineering 13 (2) (1987).

[45] G. Simmons, How to selectively broadcast a secret Proceedings of the 1985 IEEE Symposium on Security and Privacy, IEEE Computer Society, Silver Spring, MD, 1985.

[46] J. Burns, C. Mitchell, A security scheme for resource sharing over a network, Computers and Security 19 (1990) 67–76.

[47] R. Kemmerer, C. Meadows, J. Millen, Three Systems for Cryptographic Protocol Analysis 7 (2) (1994) 79–130.

[48] C. Meadows, Language generation and verification in the NRL protocol analyzer Proceedings of the 1996 IEEE Computer Security Foundation Workshop IX, IEEE Computer Society, Silver Spring, MD, 1996 pp. 48–61.

[49] CCITT CCITT X.509, The Directory—An Authentication Framework, 1988.

[50] J. Millen, C. Neuman, J. Schiller, J. Saltzer, Kerberos Authentication and Authorization system, Project Athena Technical Plan, Section E.2.1, M.I.T., MA, 1987.

[51] D. Otway, O. Rees, Efficient and timely mutual authentication, ACM Operating Systems Review 21 (1) (1987) 8–10.

[52] G. Pal, Verification of the iKP family of secure electronic payment protocols, 1996, http://web.mit.edu/gnpal/www/ikp/verify_ikp.html.

[53] S.P. Shieh, W.H. Yang, An authentication and key distribution system for open network systems, ACM Operating Systems Review 30 (2) (1996) 32–41.

[54] A. Liebl, Authentication in distributed systems: a bibliography, ACM Operating Systems Review 27 (4) (1993) 31–41.

[55] D. Nessett, A critique of the BAN logic, ACM Operating Systems Review 24 (2) (1990) 35–38.

[56] E. Snekkenes, Exploring the BAN approach to protocol analysis Proceedings of the IEEE Computer Security Foundations Workshop IV, IEEE Computer Society, Silver Spring, MD, 1991 pp. 171–181.

[57] W. Mao, C. Boyd, Towards formal analysis of security protocols Proceedings of the 1993 IEEE Computer Security Foundations Workshop VI, IEEE Computer Society, Silver Spring, MD, 1993 pp. 147–158.

[58] W. Mao, An augmentation of BAN-like logics Proceedings of the 1995 IEEE Computer Security Foundations Workshop VIII, IEEE Computer Society, Silver Spring, MD, 1995 pp. 44–56.

[59] P. Syverson, Relating two models of computation for security protocols, in: Proceedings of the 1998 LICS Workshop on Formal Methods and Security Protocols, 1998, http://www.cs.bell-labs.com/who/nch/fmsp/program.html.
[60] P.C. van Oorschot, Extending cryptographic logics of belief to key agreement protocols Proceedings of the First ACM Conference on Computer and Communications Security, ACM, New York, 1993 pp. 232–243.
[61] R. Anderson, A second generation wallet, ESORICS'92 Proceedings of the Second European Symposium on Research in Computer Security, Springer, Berlin, 1992 pp. 411–418.
[62] M. Gordon, T. Melham, Introduction to HOL: A Theorem Proving Environment for Higher Order Logic, Cambridge University Press, Cambridge, 1993.
[63] M. Abadi, M. Tuttle, A semantics for a logic of authentication Proceedings of the Tenth ACM Symposium on Principles of Distributed Computing, ACM, New York, 1991 pp. 201–216.
[64] R. Kailar, Reasoning about accountability in protocols for electronic commerce Proceedings of the 1995 IEEE Symposium on Security and Privacy, IEEE Computer Society, Silver Spring, MD, 1995 pp. 236–250.
[65] V. Kessler, H. Neumann, A sound logic for analysing electronic commerce protocols, ESORICS'98 Proceedings of the Fifth European Symposium on Research in Computer Security, Springer, Berlin, 1998 pp. 345–360.
[66] G. Wedel, V. Kessler, Formal semantics for authentication logics, ESORICS'96 Proceedings of the Fourth European Symposium on Research in Computer Security, Springer, Berlin, 1996 pp. 219–241.
[67] B. Barras, S. Boutin, C. Cornes, J. Courant, J.C. Filliatre, E. Gimenec, H. Herbelin, G. Huet, P. Manoury, C. Munoz, C. Murthy, C. Parent, C. Paulin-Mohring, A. Saibi, B. Werner, The Coq Proof Assistant Reference Manual, version 6.1., Project Coq, INRIA-Rocqunecourt and CNRS-Ens Lyon, 1996.
[68] L. Paulson, Isabelle: A Generic Theorem Prover Proceedings LNCS 828, Springer, Berlin, 1994.
[69] S. Schneider, Verifying authentication protocols with CSP Proceedings of the IEEE Computer Security Foundations Workshop X, IEEE Computer Society, Silver Spring, MD, 1997 pp. 3–17.
[70] S. Schneider, Formal analysis of a non-repudiation protocol Proceedings of the IEEE Computer Security Foundations Workshop XI, IEEE Computer Society, Silver Spring, MD, 1998 pp. 54–65.
[71] J. Zhou, D. Gollmann, A fair non-repudiation protocol Proceedings of the 1996 IEEE Symposium on Security and Privacy, IEEE Computer Society, Silver Spring, MD, 1996 pp. 55–61.
[72] F. Fabrega, J. Herzog, J. Guttman, Strand spaces: why is a security protocol correct Proceedings of the 1998 IEEE Symposium on Security and Privacy, IEEE Computer Society, Silver Spring, MD, 1998 pp. 160–171.
[73] F. Fabrega, J. Herzog, J. Guttman, Honest ideals on strand spaces Proceedings of the IEEE Computer Security Foundations Workshop XI, IEEE Computer Society, Silver Spring, MD, 1998 pp. 66–77.
[74] F. Fabrega, J. Herzog, J. Guttman, Strand space pictures, in: Proceedings of the 1998 Workshop on Formal Methods and Security Protocols, 1998, http://www.cs.bell-labs.com/who/nch/fmsp/program.html.
[75] S. Brackin, An interface specification language for automatically analyzing cryptographic protocols Proceedings of the 1997 Symposium on Network and Distributed System Security, IEEE Computer Society, Silver Spring, MD, 1997 pp. 40–51.
[76] S. Brackin, Automatic Formal Analyses of Cryptographic Protocols, Arca Systems, Inc., 1997, http://www.arca.com/paper.htm.
[77] S. Brackin, Automatic formal analyses of cryptographic protocols Proceedings of the 19th National Conference on Information Systems Security, MD, IEEE, New York, 1996.
[78] R. Lichota, G. Hammonds, S. Brackin, Verifying the correctness of cryptographic protocols using convince Proceedings of the 12th IEEE Computer Security Applications Conference, IEEE Computer Society, Silver Spring, MD, 1996 pp. 117–128.
[79] S. Brackin, Evaluating and improving protocol analysis by automatic proof Proceedings of the IEEE Computer Security Foundations Workshop XI, IEEE, New York, 1998 pp. 138–152.
[80] J. Millen, CAPSL—Common Authentication Protocol Specification Language, 1997, http://www.mitre.org/research/capsl.
[81] G. Lowe, Casper: A Compiler for the Analysis of Security Protocols, 1996, http://www.mcs.le.ac.uk/7sim;glowe/Security/Casper/index.html.
[82] M. Abadi, A. Gordon, A calculus for cryptographic protocols: the Spi calculus Proceedings of the Fourth ACM Conference on Computers and Communications Security, ACM, New York, 1997 pp. 36–47.
[83] R. Milner, J. Parrow, D. Walker, A calculus of mobile processes, Information and Computation 00 (1992) 1–77.
[84] S. Brackin, A HOL formalisation of CAPSL semantics, in: Proceedings of the 21st National Information Systems Security Conference, 1998.
[85] Programming satan's computer, in: R. Anderson, R. Needham (Eds.), Computer Science Today: Recent Trends and Developments, LNCS 1000, Springer, Berlin, 1995 pp. 426–440.
[86] M. Abadi, R. Needham, Prudent engineering practice for cryptographic protocols Proceedings of the 1994 IEEE Symposium on Security and Privacy, IEEE Computer Society, Silver Spring, MD, 1994 pp. 122–136.
[87] P. Syverson, Limitations on design principles for public key protocols Proceedings of the 1996 IEEE Symposium on Security and Privacy, IEEE Computer Society, Silver Spring, MD, 1996 pp. 62–72.
[88] C. Boyd, Towards extensional goals in authentication protocols, in: Proceedings of the 1997 DIMACS Workshop on Design and Formal Verification of Security Protocols, 1997, http://dimacs.rutgers.edu/Workshops/Security.
[89] Panel: Y. Desmedt (Chair), M. Burmester, J. Millen, Design vs. Verification: Is Verification the Wrong Approach?, in: Proceedings of the 1997 DIMACS Workshop on Design and Formal Verification of Security Protocols, 1997, http://dimacs.rutgers.edu/Workshops/Security/.
[90] C. Meadows, Formal verification of cryptographic protocols: a survey, advances in cryptology Proceedings of ASIACRYPT'94, Springer, Berlin, 1995 pp. 133–150.
[91] N. Heintze, J. Tygar, A model for secure protocols and their compositions Proceedings of the 1994 IEEE Symposium on Security and Privacy, IEEE Computer Society, Silver Spring, MD, 1994.
[92] L. Gong, P. Syverson, Fail-stop protocols: an approach to designing secure protocols, in: pre-Proceedings of DCCA-5 Fifth International Working Conference on Dependable Computing for Critical Applications, 1995, pp. 45–55.
[93] R.D. Schlichting, F.B. Schneider, Fail-stop processors: an approach to designing fault-tolerant computing systems, ACM Transactions on Computing Systems 2 (2) (1983) 222–238.
[94] U. Carlsen, Generating formal cryptographic protocol specifications Proceedings of the 1994 IEEE Computer Society Symposium on Research in Security and Privacy, IEEE Computer Society, Silver Spring, MD, 1994 pp. 137–146.
[95] C. Rudolph, A formal model for systematic design of key establishment protocols, ACISP'98 Proceedings of the Third Australasian Conference on Information Security and Privacy, Springer, Berlin, 1998 pp. 332–343.
[96] L. Buttyan, S. Staamann, U. Wilhelm, A simple logic for authentication protocol design Proceedings of the IEEE Computer Security Foundations Workshop XI, IEEE Computer Society, Silver Spring, MD, 1998 pp. 153–162.
[97] C. Boyd, W. Mao, Designing secure key exchange protocols, ESORICS'94 Proceedings of the Third European Symposium

on Research in Computer Security, Springer, Berlin, 1994 pp. 93–105.

[98] D. Gollmann, What do we mean by entity authentication? Proceedings of the 1996 IEEE Symposium on Security and Privacy, IEEE Computer Society, Silver Spring, MD, 1996 pp. 46–54.

[99] R. DeMillos, R. Lipton, A. Perlis, Social processes and proofs of theorems and programs Proceedings of the Fourth ACM Symposium on Principles of Programming Languages, ACM, New York, 1977.

[100] L. Gong, T. Lomas, R. Needham, J. Saltzer, Protecting poorly chosen secrets from guessing attacks, IEEE Journal on Selected Areas in Communications 11 (5) (1993) 648–656.

[101] S. Brackin, Automatic formal analyses of two large commercial protocols, in: Proceedings of the 1997 DIMACS Workshop on Design and Formal Verification of Security Protocols, 1997, http://dimacs.rutgers.edu/Workshops/Security/.

[102] J. Mitchell, V. Shmatikov, U. Stern, Finite-state analysis of SSL 3.0 and related protocols, in: Proceedings of the 1997 DIMACS Workshop on Design and Formal Verification of Security Protocols, 1997, http://dimacs.rutgers.edu/Workshops/Security/.

[103] L. Paulson, Inductive Analysis of the Internet Protocol TLS, TR440, University of Cambridge, Computer Laboratory, 1998.

[104] C. Bella, L. Paulson, Using Isabelle to prove properties of the Kerberos authentication system, in: Proceedings of the 1997 DIMACS Workshop on Design and Formal Verification of Security Protocols, 1997, http://dimacs.rutgers.edu/Workshops/Security/.

[105] G. Bella, L. Paulson, Kerberos version IV: inductive analysis of the secrecy goals, ESORICS'98 Proceedings of the Fifth European Symposium on Research in Computer Security, Springer, Berlin, 1998 pp. 361–375.

[106] D. Bolignano, Towards the formal verification of electronic commerce protocols Proceedings of the IEEE Computer Security Foundations Workshop X, IEEE Computer Society, Silver Spring, MD, 1997 pp. 133–146.

[107] C. Meadows, Using the NRL protocol analyzer to examine protocol suites, in: Proceedings of the 1998 LICS Workshop on Formal Methods and Security Protocols, 1998, http://www.cs.bell-labs.com/who/nch/fmsp/program.html.

[108] D. Harkins, D. Carrel, The Internet Key Exchange (IKE) version 6, 1998, ⟨draft-ietf-ipsec-isakmp-oakley-06.txt⟩.

Dr. Stefanos Gritzalis *holds a BSc (Physics), an MSc (Electronic Automation), and a PhD (Informatics) all from the University of Athens, Greece. Currently, he is an Assistant Professor with the Department of Informatics of the Technological Educational Institute (TEI) of Athens, Greece, and a Senior Researcher with the Department of Information and Communication Systems of the University of Aegean, Greece. His professional experience includes senior consulting and researcher positions in a number of private and public institutions. He has been involved in several national and EU funded R and D projects, in the areas of Computer Security and Distributed Systems. His published scientific work includes two books (in Greek) on Information and Communication Technologies topics, and more than 20 journal and national and international conference papers. The focus of these publications is on Computer Systems Security, Cryptography, and Distributed Systems. Dr. S. Gritzalis is a Member of the Board (Treasurer) of the Greek Computer Society.*

Dr. Diomidis Spinellis *holds an MEng in Software Engineering and a PhD in Computer Science both from Imperial College (University of London, UK). Currently, he is lecturing at the Department of Information and Communication Systems, University of Aegean, Greece. He has provided consulting services to a number of Greek and international Information Technology companies. He has been involved in several national and EU funded R and D projects in the areas of Computer Security, Language Engineering, and Scientific Visualisation. He is the author of more than 40 technical papers and conference presentations. He has contributed software to the 4.4BSD Unix distribution, the X-Windows system, and is the author of a number of public domain software packages, libraries, and tools. His research interests include Information Security, Software Engineering, and Programming Languages. Dr. Spinellis is a member of the ACM, the IEEE, and a founding member of the Greek Internet User's Society. He is a co-recipient of the Usenix Association 1993 Lifetime Achievement Award.*

Dr. Panagiotis Georgiadis *holds a BSc (Physics) from the University of Athens, Greece, an MSc in Computer Science from the Warwick University, UK, and a PhD in Computer Science from the University of Athens, Greece. Currently, he is an Associate Professor with the Department of Informatics, University of Athens, Greece. He has been involved with several funded R and D projects in the areas of Distributed Systems and Software Engineering. He has authored three books and more than 50 papers and conference presentations. His research interests include Simulation, Distributed Systems, Operating Systems, and Computer Systems Security. Dr. Georgiadis is a member of the ACM, and the Greek Computer Society.*