Formal methods in the design of cryptographic protocols*
(state of the art)

Levente Buttyán
Swiss Federal Institute of Technology
Institute for computer Communications and Applications (ICA)
EPFL-DI-ICA, CH-1015 Lausanne, Switzerland
Levente.Buttyan@epfl.ch

26th November 1999

Abstract

This paper is a state of the art review of the use of formal methods in the design of cryptographic protocols.

1 Introduction

Cryptographic protocols are used to provide secure services in distributed systems. However, if the protocol is not designed carefully enough, it may contain flaws, which can be the ideal starting point for various attacks. Such flaws can be subtle and hard to find. A number of examples exist in the academic literature of flaws that were not found for some time in protocols that had received intensive analysis. These examples include the Needham-Schroeder authenticated key distribution protocol [NS78], which was found by Denning and Sacco to allow an intruder to pass an old, compromised session key as a new one to a legitimate party [DS81]; a protocol in the CCITT X.509 draft standard [CCITT], for which Burrows, Abadi, and Needham showed that an intruder can cause an old session key to be accepted as a new one, whether or not it had been compromised [BAN90a]; and many others.

These examples show that the informal design of cryptographic protocols is error prone. Formal methods seem to be better suited for solving the problem. Formal methods have long been used in the design and analysis of communication protocols in general. From the early 90's, they are widespreadly used in the field of cryptographic protocols too. Formal techniques can be used in various phases of the design of a cryptographic protocol. These phases include the specification, the construction, and the verification. Up to now, most of the work was concentrated on the formal verification of cryptographic protocols. There

*EPFL SSC Technical Report No. SSC/1999/038

are also some notable work on the formal specification of protocols. However, using formal methods to construct protocols in a systematic way is an undiscovered area yet. In the sequel, we give an overview of the most significant results in the formal specification, construction, and verification of cryptographic protocols. More thorough surveys can be found in [RH93], [Mea95], and [GNG97].

2 Specification

The design of a cryptographic protocol begins with the specification of the requirements that the designer wants the protocol to satisfy. This means that the designer should have a clear idea of what he/she wants the protocol to achieve, thus, what the correctness of the protocol means. However, expressing the correctness criteria of a protocol is not trivial. Early work on this field concentrated on secrecy as the main correctness criterion. Later, it was realised to be inadequate, since many cryptographic protocols provide services, such as authentication, that are only indirectly related to secrecy. This problem has been approached from several different angles, some with the aim of developing a set of criteria that can be applied to protocols in general, and others with the aim of developing ways to express criteria for a number of different types of protocols.

In [DvW92], Diffie *et. al.* present informal requirements for the correctness of an authentication protocol. Briefly, they say that session keys should remain secret and that protocol runs should match. The latter means that if A and B run a protocol then A's record of messages received from B should match B's record of messages sent to A, and vica versa. This notion has been formalised by Bellare and Rogaway in [BR94], using a model based on communicating probabilistic Turing machines. The notion of matching runs has also been formalised by Syverson in his extension of the Abadi-Tuttle logic to include temporal formalisms [Syv93]. In [WL93], Woo and Lam take a similar approach to defining security of authentication protocols. They define a semantic model for authentication protocols that is based on two basic security properties, correspondence and secrecy. The former requires that certain events can take place only if others have taken place previously. This is very similar to the notion of matching runs, but it is broader, since the events do not have to be the sending and the receiving of the same message.

Another approach to specifying requirements for protocols is presented in the requirements language developed for the NRL Protocol Analyser [SM93]. The requirements specified in this language have a form similar to the notion of correspondence of Woo and Lam, in that the requirements are given on sequences of events. The difference is that, instead of giving general requirements for correspondence that applies to all protocols, the user of the language can specify requirements for protocols of a particular class. Thus, requirements can vary according to the intended function of the protocol. Furthermore, the action, by which the intruder learns a word is modelled as an event, thus, secrecy does not need to be defined as a separate part of the model. In [SM93], Syverson and Meadows give a set of requirements for various kinds of message authentication protocols, while in [SM95], they give requirements for key distribution protocols with repeated authentication.

Specifying requirements for cryptographic protocols in a formal way can be useful in understanding the problems that the designer of the protocol has to solve, and in bridging the abstraction gap between informal correctness criteria and the formal protocol description. Therefore, it helps the design of cryptographic protocols.

3 Construction

Most of the existing work in the application of formal methods to cryptographic protocols has been concentrated on the formal verification of existing protocols. However, it would be cheaper and more effective to use these methods in the design of the protocol in the first place, and so save the expense of redesign. Although, the use of formal methods in the design is a natural application of the technology, not much research has been done in this particular area.

One approach to incorporate formal methods into the design is to develop specific methodologies for design of protocols so that they will be more amenable to analysis by formal methods. This approach is taken by Heintze and Tygar in [HT94]. They develop a modular approach to design cryptographic protocols. They design a family of tools for reasoning about protocol security and prove a composition theorem that allows them to state sufficient conditions on two secure protocols, such that they may be combined to form a new secure protocol. They give counter-examples to show that when the conditions are not met, the new protocol may not be secure.

In [GS95], Gong and Syverson present a new methodology to facilitate the design and analysis of secure cryptographic protocols. They propose to restrict protocol designs to well-defined practices, instead of ever increasing the complexity of protocol security analysis mechanisms to deal with every newly discovered attack and the endless variations in protocol construction. In particular, they introduce a novel notion of a fail-stop protocol, which automatically halts in response to any active attack, thus reducing protocol security analysis to that of passive attacks only. Excluding the possibility of an active attack makes the validation of the paradoxical secrecy assumption in BAN-like logics easier, and thus, puts modal logic based analysis methods on a stronger footing. Gong and Syverson suggest types of protocols that are fail-stop, however, their suggestion might not be practical for some applications. In [KS96], Keromytis and Smith present a generic method for creating efficient fail-stop cryptographic protocols.

An alternative approach to use formal methods in the design of cryptographic protocols is a layered one [Mea95], in which a relatively abstract model is used at the top layer, and each succeeding layer is proved to be an implementation (refinement) of the layer above it, until finally either a detailed specification or the actual protocol code is produced. This approach is taken by the recent author *et. al.* in [BSW98]. They propose a BAN-like logic extended with the notion of channels with various access restrictions that can be used as the abstract model at the top layer. In particular, based on the logic they construct synthetic rules that they use to derive high level protocol descriptions from the goals of the protocol. A similar method on a stronger footing is described in [AFS97] by Alves-Foss and Soule, although they do not actually use their results for deriving protocols.

In [BM94], Boyd and Mao propose a technique to design key exchange protocols that are guaranteed to be correct in the sense that a specified security criterion will not be violated if the protocol participants act correctly. This technique is developed from basic cryptographic properties that are 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.

Finally, we mention [AN94], in which Abadi and Needham present principles for designing cryptographic protocols. Their principles are informal guidelines, but they can complement formal methods. The principles presented are neither necessary nor sufficient for correctness of protocols, however, they are helpful in that adherence to them would have prevented a number of published errors. In [Syv96], Syverson criticises this approach. He presents limitations and exceptions on some of the basic design principles, and gives examples for secure protocols that fail to meet the principles.

4 Verification

In [Mea95], Meadows classifies approaches to the formal analysis of cryptographic protocols into four types:

- Modelling and verifying the protocol using specification languages and verification tools not specifically developed for the analysis of cryptographic protocols.
- Developing expert systems that a protocol designer can use to investigate different scenarios.
- Modelling and verifying the protocol using modal logics developed for the analysis of knowledge and belief.
- Developing a formal model based on the algebraic term-rewriting properties of cryptographic systems.

In the sequel, we follow this classification. Since we are interested in the logic based design of security services, we put the emphasis on the modal logic based approach.

4.1 Using general purpose verification tools

The main idea of this approach is to treat a cryptographic protocol as any other (distributed) program and attempt to prove its correctness. The first step is to specify the protocol and its correctness requirements so that the techniques apply. For this purpose, Varadharajan uses LOTOS [Var90], Kemmerer specifies the system in Ina Jo [Kem89], while others use even more general description techniques such as state machines [Var89] or Petri nets [NT92]. Once the protocol and its requirements are specified, it can be investigated by using the tools that are available in the formalism used.

In [Var90], Varadharajan proposes the use of LOTOS to analyse authentication protocols. He gives example specifications of protocols in LOTOS, but he cannot demonstrate any result in their analysis. The paper concludes by stating that LOTOS tools are not yet adequate for this kind of analysis.

In [Kem89] and [KMM94], Kemmerer uses an extension of first-order predicate calculus, a formal specification language called Ina Jo. Ina Jo was designed as a general purpose tool to support software development and correctness proofs. Kemmerer describes an example security system, and then gives an Ina Jo specification of it. He also specifies critical requirements that the system is to satisfy in all states. Once the specification is complete, Ina Jo generates theorems that can be used to verify if the critical requirements are satisfied. Kemmerer uncovers a weakness in his sample system, but the value of this method is limited, because the specification of the critical requirements needs that the designer knows the potential attacks in advance.

In [Var89], Varadharajan describes how to specify a protocol using state diagrams. Each party of the protocol is described by a state diagram, and then the protocol is represented as the cross product of the state diagrams for each individual parties. Once the protocol is described in this way, the designer can investigate various executions of the protocol by applying a technique known as the reachability analysis. Reachability analysis techniques are effective in determining whether or not a protocol is correct with respect to its specification, but they do not guarantee resistance against an active intruder. This would require to model the intruder, which, in general, leads to the quick growth of the number of the states (state explosion problem).

In [NT92], Nieh and Tavares uses a Petri net based methodology for the formal modelling and analysis of cryptographic protocols. In particular, they use coloured Petri nets to model protocols. Their model also includes a general intruder model that can be used to formulate intruder attacks and generate test cases. The analysis of the security properties of cryptographic protocols is based on an exhaustive penetration test that searches for scenarios that violate certain specified criteria. These criteria are defined in terms of requirements on Petri net states of the protocol. Although, coloured Petri nets enable to produce compact and manageable protocol descriptions, tools that support the effective execution of an exhaustive search are still missing. The designer can translate these high level Petri nets into ordinary Petri nets, and can use tools available for those, but then he/she has to cope with the state explosion problem again.

A more recent approach models the protocol participants as Communicating Sequential Processes (CSP) and uses the Failure Divergence's Refinement (FDR) checker, which is a general purpose tool that can be used to determine whether an implementation refines a specification. First FDR has been used to analyse many sorts of systems, including distributed databases and communication protocols, but recently it has been used to analyse security protocols as well [Ros95] [Low96]. Another recent approach is to use the Higher Order Logic (HOL) for stating and proving properties of cryptographic protocols [Sne95]. Tools that are based on HOL and used for analysing cryptographic protocols are include Convince [LHB96] and Isabelle [BP97].

Although, there are some notable attempts to use general purpose verification tools for analysing cryptographic protocols, they are, in general, not suitable for this purpose. The main weakness of this approach is that in applying methods that were not intended specifically for security analysis, subtle pitfalls that are peculiar to the security domain are not considered. Furthermore, including the intruder with full generality into the model leads to the state explosion problem and makes the analysis infeasible.

4.2 Developing expert systems

The idea of this approach is to develop expert systems that the protocol designer can use to generate and investigate various scenarios. Most of these systems are based on an underlying, state machine based model of the protocol. But, as opposed to the state machine based analysis of cryptographic protocols described in the previous subsection, these systems begin with an undesirable state and attempt to discover if this state is reachable from an initial state.

One of the earliest system of this approach is the Interrogator by Millen *et. al.* [MCF87]. In the Interrogator, protocol participants are modelled as communicating state machines whose messages to each other are intercepted by the 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 tries all possible ways to construct a path by which that state can be reached. If it finds such a path, then a security flaw is identified. The Interrogator has not yet found a previously unknown attack on a cryptographic protocol, but it has been able to reproduce a number of known attacks [KMM94].

The NRL Protocol Analyzer by Meadows [KMM94] is similar to the Interrogator: the designer specifies an insecure state and the Analyzer attempts to construct a path to that state from an initial state. Unlike the Interrogator, an unlimited number of protocol rounds are allowed in a single path. This allows the Analyzer to discover attacks that rely on the intruder's ability to weave several different runs of a protocol together. Also unlike the Interrogator, the emphasis is, not only on finding paths to insecure states, but on proving that these states are unreachable. This is made possible by having the user prove that certain paths leading backwards from an insecure state go into infinite loops, thus never reach an initial state. Once these paths have been eliminated, the resulting search space is often small enough to search exhaustively. The proofs that paths lead into infinite loops are largely guided by the user, thus, the search is less automated than in the Interrogator. The NRL Protocol Analyzer has been successful in finding several previously unknown security flaws in cryptographic protocols [Mea91] [Mea92].

In [LR92], Longley and Rigby use a rule based system that transforms goals into subgoals and can constantly continue this process. They use this rule based scheme to build a tree, in which each node represents a data item, and the children of a node represent those data items that are required for the knowledge of the data represented by the father node. In this way, they can construct a tree, in which the root node represents the data required by the intruder for an attack (e.g., a cryptographic key), and the leaves represent those data items that are required to know the root item. The Longley and Rigby tool allows the user to interact with the system. The user can determine whether a data can or cannot be found by the intruder. If the data is judged to be accessible, this information can be inserted into the system, and the generation of the tree can proceed. Longley and Rigby managed to find a subtle and previously unknown flaw in a hierarchical key management scheme.

Expert systems developed specifically for the analysis of cryptographic protocols are more successful than general purpose tools. However, they are often inefficient because they perform exhaustive search. Sometimes they do not even halt and the results are inconclusive. To cope with these problems, they require human intervention. Their advantage is that if they discover a flaw, then the attack scenario that exploits the flaw is directly available, which is not the case with the more popular modal logic based approach described in the next subsection.

4.3 Modal logic based approach

The main idea of this approach is to use modal logics, similar to those that has been developed for the analysis of the evolution of knowledge and belief during the execution of distributed algorithms [HM90], for modelling and analysing cryptographic protocols. After all, cryptographic protocols can be viewed as distributed algorithms. Such logics consists of a language, which is used to describe various statements about the protocol such as what the participants know or believe, and some inference rules, which are used to derive new statements from previously derived statements. The goal of the analysis is to derive a statement that represents the correctness condition of the protocol. The designer's inability to do so means that the protocol may not be correct. The analysis often reveals the possible weakness in the protocol, and an attack can be constructed easily afterwards.

The best known and most influential logic of this type is called BAN logic [BAN90b]. It can be used to describe the beliefs of trustworthy parties involved in authentication protocols and the evolution of these beliefs as a consequence of communication. It has been successfully applied to discover flaws in a variety of authentication protocols and has also been helpful in the understanding of the basic concepts of authentication. BAN logic is simple, which might be one of the reasons for its popularity. The need for many universal assumptions in the underlying model, however, is a minor disadvantage. In BAN logic, it is assumed that principals of the system are trustworthy and do not release secrets (this has not always been understood with its full meaning [Nes90]); the applied encryption schemes are strong (i.e., encrypted messages can be decrypted only with the appropriate key); each encrypted message contains sufficient redundancy to allow a principal who decrypts it to verify that it has used the right key; and principals can recognise and ignore their own messages.

BAN logic does not attempt to model a protocol in a richness as other logics do. It does not attempt to model the distinction between seeing a message and understanding it; it does not attempt to model the revision of beliefs; it does not attempt to model trust or the lack of it; and finally, it does not attempt to model knowledge. The avoidance of these issues is intentional in BAN logic; this makes it simple and straightforward. However, this also means that these issues have to be addressed in the informal mapping from protocol specification to BAN specification. This mapping is called *idealisation*. Burrows *et. al.* consider the idealised protocols as clearer and more complete specifications than the traditional descriptions found in the literature, which they view as implementation dependent encodings of the protocol. Although, this is true, no clear idealisation method is presented, which led to misunderstandings and the misuse of the logic [Nes90] [BM93].

To overcome these problems, Abadi and Tuttle reformulate the original BAN logic and provide a new semantics for it in [AT91]. Abadi and Tuttle identifies many sources of the confusion created by BAN logic. They remove some unnecessary mixing of semantic and implementation details in the original definitions and inference rules. They define all concepts, such as seeing, saying, believing, etc., independently rather than jointly with other concepts. They reformulate the set of inference rules as an axiomatisation with modus ponens and necessitation as the only rules. These changes make the logic much simpler and allow them to dispense with an implicit assumption of honesty (it is not needed anymore that principals believe the messages they send), which is not well-defined in general. One of the greatest contribution of the paper is a formal semantics, which defines belief as a form of resource-bounded, defeasible knowledge. Abadi and Tuttle claim that the reformulated logic is sound according to the new semantics defined. However, Syverson and van Oorschot point out in [SvO94] that one of the axioms is not sound. Abadi and Tuttle are still working on a revision of their paper[1].

BAN logic has been extended in many directions. In [GNY90], Gong *et. al.* propose a logic referenced later as GNY logic. In GNY logic, it is not required to assume that principals are trustworthy and redundancy is always explicitly present in encrypted messages. GNY logic distinguishes between what a principal can possess and what it can believe in. It enables to express different trust levels and implicit conditions behind protocol steps. This makes GNY logic a more realistic model of cryptographic protocols than BAN logic. However, GNY logic has more than 40 inference rules, which has led many to reject this approach as being impractical.

Other extensions of BAN logic include [MB93], in which Mao and Boyd propose a logic with several improvements on the original BAN logic; [GS91], in which Gaarder and Snekkenes extend BAN logic with constructs to reason about time; [van93], in which van Oorschot extends BAN and GNY to deal with key agreement protocols such as Diffie-Hellman; and [CSNP92], in which Cambell *et. al.* extend BAN logic using probabilistic reasoning to calculate a measure of trust rather than complete trust. The list of extensions above is not complete; there are many other works in this field.

In [SvO94], Syverson and van Oorschot propose a logic, later referenced as SvO, that encompasses a unification of four of its predecessors in the BAN family, namely GNY logic [GNY90], the Abadi-Tuttle logic [AT91], the logic proposed by van Oorschot in [van93], and BAN itself [BAN90b]. The proposed logic captures all of the desirable features of its predecessors, nonetheless, it accomplishes this with no more axioms or rules than the simplest of its predecessors. Syverson and van Oorschot also present a model-theoretic semantics with respect to which the logic is sound.

The importance of having an alternative, independently motivated semantics is emphasised by Syverson in [Syv90], [Syv91b], [Syv91a], and [Syv92]. A formal semantics provides a precise structure with respect to which soundness and completeness of the logic may be proven, and thus, which allows us to evaluate the logic. If a logic does not have an independently motivated semantics, then whatever assurances protocol analysis via such logic brings may prove illusory. Syverson also illustrates how the semantics itself can be used as a reasoning tool to discover results that would be very difficult or impossible to prove by reasoning syntactically.

There are a number of other logics that do not belong to the BAN family. These include Bieber's CKT5 [Bie90], Syverson's KPL [Syv90], Moser's logic [Mos89], Rangan's logic [Ran88], and the system of Yahalom *et. al.* [YKB93]. Bieber's CKT5 and Syverson's KPL can be used to reason about the evolution of knowledge about words used in a cryptographic protocol. They also make a distinction between seeing a message and understanding its significance. Moser's logic is particular, because it is the only non-monotonic logic considered here. It can be used to reason about beliefs of protocol participants, and about how these

[1]Personal communication.

beliefs change if, for instance, a key used in a secure communication becomes compromised. Rangan's logic can be used to reason about the effect of trust in the composition of secure communications channels, and provides a formal basis for the evolution of belief from trust. The system of Yahalom *et. al.* is used to derive information about the nature of the trust that parties in a protocol must have in each other in order a protocol to operate correctly.

Using modal logics of knowledge and belief is the most popular approach to apply formal methods in the analysis of cryptographic protocols. The reason is, probably, that these logics are easy to use and intuitive. However, one must be careful, because there are often subtle assumptions in the underlying model, getting away from which may lead to the misuse of the logic and errors in the analysis. Thus, the designer has to know the scope of the logic applied and has to be aware of its limitations.

4.4 Algebraic approach

Another approach to apply formal methods to cryptographic protocol analysis is to model the protocol as an algebraic system. Research in this area has not been as active as research in developing logics of belief and knowledge. However, algebraic models was successful to represent very subtle kinds of knowledge in cryptographic protocols. Furthermore, the fact that the objects modelled correspond strongly to entities and messages used in the tools based on state machines suggests that algebraic models could be used to provide the state machine tools with a stronger capability of modelling the knowledge that the intruder can gain [Mea95]. However, the question of how these algebras can be incorporated in state machine based analysis tools is still open.

The first work that considers a cryptographic protocol to be an algebraic system is [DY83] by Dolev and Yao. In their model, Dolev and Yao assume that the network is under the control of the intruder who can read all traffic, alter and destroy messages, and perform any operation, such as encryption, that is available to legitimate users of the system. However, it is assumed that initially the intruder does not know any information that is to be kept secret, such as encryption keys belonging to legitimate users of the system. Since the intruder can prevent any message from reaching its destination, and since he/she can also create messages of his/her own, Dolev and Yao treat any message sent by a legitimate user as a message sent to the intruder and any message received by a legitimate user as a message received from the intruder. Thus, the system becomes a machine used by the intruder to generate words. These words obey certain rewrite rules, such as the fact that encryption and decryption with the same key cancel each other out. Thus, finally, the intruder manipulates a term rewriting system. If the goal of the intruder is to find out a word that is meant to be secret, then the problem of proving a protocol secure is equivalent to the problem of proving that a certain word cannot be generated in a term rewriting system. Dolev and Yao uses this idea to investigate the security of certain classes of public key protocols. They define the notion of *cascade protocols* and *name-stamp protocols*. They give sufficient and necessary conditions for cascade protocols to be secure, and provide a polynomial time algorithm to decide if a given name-stamp protocol is secure.

The Dolev-Yao model is too restricted to be useful for the analysis of most protocols. It can only be used to detect failures of secrecy, and it does not allow participants to remember state information from one state to the next. Thus, most later works extend it to be capable for analysing other classes of protocols. These works include [Mer83], in which Merritt generalises the technique of Dolev and Yao to model diverse cryptographic systems and to formally state and prove security properties other than secrecy. In [Tou92], Toussaint describes a technique for deriving the complete knowledge of participants in cryptographic protocols that is based on the algebraic model of Merritt.

A more recent work is [AG98], in which Abadi and Gordon use the *pi* calculus to describe protocols at an abstract level. They use the *pi* calculus primitives for channels, in particular the scoping rules, to model certain properties of cryptographic protocols. Then, they extend the *pi* calculus to what they call *spi* calculus to analyse protocols at a less abstract level. The *spi* calculus permits an explicit representation of the use of cryptography in protocols. In the *spi* calculus, security properties of the protocol are expressed as equivalences between *spi* calculus processes (e.g., a protocol keeps secret a piece of data X if the protocol with X is equivalent to the protocol with X' for every X'). The intruder is not explicitly modelled, and this is a main advantage of the approach. Modelling the intruder can be tedious and can lead to errors (e.g., it is very difficult to model that the intruder can invent random numbers but is not lucky enough to guess the random secrets on which the protocol depends). Instead, the intruder is represented as an arbitrary *spi* calculus process.

5 Conclusion

We gave an overview of the recent approaches to use formal methods in the design of cryptographic protocols. Formal methods can be used in various phases of the design, such as specification, construction, and verification. Up to now, most of the work concentrated on the formal verification of protocols, and in particular, on the application of modal logics of knowledge and belief for modelling and analysing protocols. Using formal methods in the specification of cryptographic protocols is an emerging area of research, while formal methods to construct protocols in a systematic way are just appearing.

References

[AFS97] J. Alves-Foss and T. Soule. A weakest precondition calculus for analysis of cryptographic protocols. In *Proceedings of the DIMACS Workshop on Design and Formal Verifictaion of Security Protocols*, 1997.

[AG98] M. Abadi and A. Gordon. A calculus for cryptographic protocols: The Spi calculus. Technical Report SRC RR 149, Digital Equipment Corporation, Systems Research Center, January 1998.

[AN94] M. Abadi and R. Needham. Prudent engineering practice for cryptographic protocols. In *Proceedings of the IEEE CS Symposium on Research in Security and Privacy*, pages 122–136, 1994.

[AT91] M. Abadi and M. Tuttle. A semantics for a logic of authentication. In *Proceedings of the Tenth Annual ACM Symposium on Principles of Distributed Computing*, pages 201–216, 1991.

[BAN90a] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, February 1990.

[BAN90b] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. Technical Report SRC RR 39, Digital Equipment Corporation, Systems Research Center, February 1990.

[BGSW98] L. Buttyán, C. Gbaguidi, S. Staamann, and U. Wilhelm. A note on an authentication technique based on distributed security management for the global mobility network. Technical Report SSC/98/18, Swiss Federal Institute of Technology, April 1998.

[Bie90] P. Bieber. A logic of communication in a hostile environment. In *Proceedings of the IEEE CS Computer Security Foundations Workshop*, pages 14–22, June 1990.

[BM93] C. Boyd and W. Mao. On a limitation of BAN logic. In *Advances in Cryptology - EUROCRYPT'93*, pages 240–247, 1993.

[BM94] C. Boyd and W. Mao. Designing secure key exchange protocols. In *Proceedings of the European Symposium on Research in Computer Security*, 1994.

[BP97] G. Bella and L. Paulson. Using Isabelle to prove properties of the Kerberos authentication system. In *Proceedings of the DIMACS Workshop on Design and Formal Verifictaion of Security Protocols*, 1997.

[BR94] M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Advances in Ctyptology - CRYPTO'93*, pages 232–249. Springer-Verlag, 1994.

[BSW98] L. Buttyán, S. Staamann, and U. Wilhelm. A simple logic for authentication protocol design. In *Proceedings of the IEEE CS Computer Security Foundations Workshop*, 1998.

[CCITT] CCITT. CCITT Draft Recommendation X.509. The Directory Authentication Framework, Version 7, Nov 1987.

[CSNP92] E. Campbell, R. Safavi-Naini, and P. Pleasants. Partial belief and probabilistic reasoning in the analysis of secure protocols. In *Proceedings of the IEEE CS Computer Security Foundations Workshop*, 1992.

[DS81] D. Denning and G. Sacco. Timestamps in key distribution protocols. *Communications of the ACM*, 24(8):198–208, 1981.

[DvW92] W. Diffie, P. van Oorschot, and M. Wiener. Authentication and authenticated key exchanges. *Designs, Codes, and Cryptography*, 2:107–125, 1992.

[DY83] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, IT-29(2):198–208, March 1983.

[GNG97] S. Gritzalis, N. Nikitakos, and P. Georgiadis. Formal methods for the analysis and design of cryptographic protocols: A state-of-the-art review. In *Proceedings of the IFIP Working Conference on Communications and Multimedia Security, Vol. 3.*, pages 119–132, 1997.

[GNY90] L. Gong, R. Needham, and R. Yahalom. Reasoning about belief in cryptographic protocols. In *Proceedings of the IEEE CS Symposium on Research in Security and Privacy*, pages 234–248, 1990.

[GS91] K. Gaarder and E. Snekkenes. Applying a formal analysis technique to the CCITT X.509 strong two-way authentication protocol. *Journal of Cryptology*, 3:81–98, 1991.

[GS95] L. Gong and P. Syverson. Fail-stop protocols: A new approach to designing secure protocols. In *Proceedings of the 5th International Working Conference on Dependable Computing for Critical Applications*, pages 44–55, 1995.

[HM90] J. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, 37(3):549–587, 1990.

[HT94] N. Heintze and J. Tygar. A model for secure protocols and their compositions. In *Proceedings of the IEEE CS Symposium on Research in Security and Privacy*, pages 2–13, 1994.

[Kem89] R. Kemmerer. Analyzing encryption protocols using formal verification techniques. *IEEE Journal on Selected Areas in Communications*, 7(4):448–457, October 1989.

[KMM94] R. Kemmerer, C. Meadows, and J. Millan. Three systems for cryptographic protocol analysis. *Journal of Cryptology*, 7(2):79–130, 1994.

[KS96] A. Keromytis and J. Smith. Creating efficient fail-stop cryptographic protocols. Technical Report MS-CIS-96-32, University of Pennsylvania, December 1996.

[LABW92] B. Lampson, M. Abadi, M. Burrows, and E. Wobber. Authentication in distributed systems: Theory and practice. *ACM Transactions on Computer Systems*, 10(4):265–310, November 1992.

[LHB96] R. Lichota, G. Hammonds, and S. Brackin. Verifying the correctness of cryptographic protcols using Convince. In *Proceedings of the 12th IEEE Computer Security Applications Conference*, pages 117–128, 1996.

[Low96] G. Lowe. Breaking and fixing the Needham-Schroeder public key protocol using FDR. In *Proceedings of the TACAS*, pages 147–166, 1996.

[LR92] D. Longley and S. Rigby. An automatic search for security flaws in key management schemes. *Computers and Security*, 11(1):75–89, January 1992.

[MB93] W. Mao and C. Boyd. Towards formal analysis of security protocols. In *Proceedings of the IEEE CS Computer Security Foundations Workshop*, pages 147–158, 1993.

[MCF87] J. Millen, S. Clark, and S. Freedman. The Interrogator: Protocol security analysis. *IEEE Transactions on Software Engineering*, SE-13(2):274–288, February 1987.

[Mea91] C. Meadows. A system for the specification and analysis of key management protocols. In *Proceedings of the IEEE CS Symposium on Research in Security and Privacy*, pages 182–195, 1991.

[Mea92] C. Meadows. Applying formal methods to the analysis of a key management protocol. *Journal of Computer Security*, 1(1):5–35, 1992.

[Mea95] C. Meadows. Formal verification of cryptographic protocols: A survey. In *Advances in Cryptology - ASIACRYPT'94*, pages 135–150. Springer-Verlag, 1995.

[Mer83] M. Merritt. *Cryptographic Protocols.* PhD thesis, Georgia Institute of Technology, 1983.

[Mil97] J. Millen. Common Authentication Protocol Specification Language. http://www.mitre.org/research/capsl, 1997.

[Mos89] L. Moser. A logic of knowledge and belief for reasoning about computer security. In *Proceedings of the IEEE CS Computer Security Foundations Workshop*, pages 57–63, 1989.

[Nee97] R. Needham. The changing environment for security protocols. *IEEE Network*, pages 12–15, May/June 1997.

[Nes90] D. Nessett. A critique of the Burrows, Abadi and Needham logic. *Operating Systems Review*, 24(2):35–38, April 1990.

[NS78] R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978.

[NT92] B. Nieh and S. Tavares. Modelling and analysing cryptographic protocols using Petri nets. In *Proceedings of AUSCRYPT'92*, 1992.

[Ran88] P. Rangan. An axiomatic basis of trust in distributed systems. In *Proceedings of the IEEE CS Symposium on Research in Security and Privacy*, pages 204–211, April 1988.

[RH93] A. Rubin and P. Honeyman. Formal methods for the analysis of authentication protocols. Technical Report CITI TR 93-7, October 1993.

[Ros95] A. Roscoe. Modelling and verifying key exchange protocols using CSP and FDR. In *Proceedings of the IEEE CS Computer Security Foundations Workshop*, pages 98–107, 1995.

[SM93] P. Syverson and C. Meadows. A logical language for specifying cryptographic protocol requirements. In *Proceedings of the IEEE CS Symposium on Research in Security and Privacy*, pages 165–177, 1993.

[SM95] P. Syverson and C. Meadows. Formal requirements for key distribution protocols. In *Advances in Cryptology - EUROCRYPT'94*, pages 320–331. Springer-Verlag, 1995.

[Sne95] E. Snekkenes. *Formal Specification and Analysis of Cryptographic Protocols.* PhD thesis, University of Oslo, Norway, 1995.

[SvO94] P. Syverson and P. van Oorschot. On unifying some cryptographic protocol logics. In *Proceedings of the IEEE CS Symposium on Research in Security and Privacy*, pages 14–28, 1994.

[Syv90] P. Syverson. Formal semantics for logics of cryptographic protocols. In *Proceedings of the IEEE CS Computer Security Foundations Workshop*, pages 32–41, June 1990.

[Syv91a] P. Syverson. The use of logic in the analysis of cryptographic protocols. In *Proceedings of the IEEE CS Symposium on Research in Security and Privacy*, pages 156–170, 1991.

[Syv91b] P. Syverson. The value of semantics for the analysis of cryptographic protocols. In *Proceedings of the IEEE CS Computer Security Foundations Workshop*, pages 228–229, 1991.

[Syv92] P. Syverson. Knowledge, belief, and semantics in the analysis of cryptographic protocols. *Journal of Computer Security*, 1(3):317–334, 1992.

[Syv93] P. Syverson. Adding time to a logic of authentication. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 97–101, November 1993.

[Syv96] P. Syverson. Limitations on design principles for public key protocols. In *Proceedings of the IEEE CS Symposium on Research in Security and Privacy*, pages 62–73, 1996.

[Tou92] M-J. Toussaint. Deriving the complete knowledge of participants in cryptographic protocols. In *Advances in Cryptology - CRYPTO'91*, pages 24–43. Springer-Verlag, 1992.

[van93] P. van Oorschot. Extending cryptographic logics of belief to key agreement protocols. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 232–243, 1993.

[Var89] V. Varadharajan. Verification of network security protocols. *Computers and Security*, 8(8):693–708, 1989.

[Var90] V. Varadharajan. Use of a formal description technique in the specification of authentication protocols. *Computer Standards and Interfaces*, 9:203–215, 1990.

[WL93] T. Woo and S. Lam. A semantic model for authentication protocols. In *Proceedings of the IEEE CS Symposium on Research in Security and Privacy*, pages 178–194, 1993.

[YKB93] R. Yahalom, B. Klein, and T. Beth. Trust relationships in secure systems: A distributed authentication perspective. In *Proceedings of the IEEE CS Symposium on Research in Security and Privacy*, pages 150–164, 1993.