Formal Testing & Algebraic Modelling Techniques for Verifying Cryptographic Protocols

T. Newe, T. Coffey,
Dept. of Electronic & Computer Eng.,
University of Limerick,
Ireland.
E-Mail: (Thomas.Newe@UL.ie) (Tom.Coffey@UL.ie)
Fax: 353-61-338176

Abstract: Before trusting a communications security protocol with potentially critical or sensitive information, it is necessary to have some degree of assurance that the protocol fulfils its intended objectives. To provide this assurance it is necessary to use formal verification techniques, as intuitive reasoning does not satisfactorily guarantee complete freedom from protocol errors. In this paper, a number of recently published formal techniques for verifying cryptographic protocols, are described. The techniques are categorised as being based on algebraic modelling, or based on some form of testing. A comparative overview of the verification techniques available is presented, indicating their benefits, limitations and scope of application.

1. Introduction

Traditionally, cryptographic protocols have been designed and verified using informal and intuitive techniques. However, an absence of formal verification can lead to flaws and security errors remaining undetected in a protocol, such as the flaw in the Needham-Schroeder authentication protocol [1] presented in [2]. A number of formal techniques have recently been developed for this purpose and can be broadly divided into the following categories:

i. Testing or State Machine based Techniques.
ii. Algebraic modelling.
iii. Logical techniques, based on the modal logics of belief and/or knowledge.

This paper reviews state machine and algebraic modelling techniques. The fundamental characteristics of the techniques are described and their limitations highlighted. The scope of application for these techniques is also explained.

2. Testing/State Machine Techniques

Testing techniques model cryptographic protocols as a set of finite-state machines. The security of the protocol is validated, either by an exhaustive search of the state space (looking for a possible weak state which allows an intruder access to some secret information [3]), or by the use of proof techniques for reasoning about state machine models [4]. Generally, some form of specialised test software is required. Most testing techniques embody some of the aspects of the work of Dolev and Yao [5,6]. While the Dolev and Yao

model is solely designed to detect failures in secrecy, their work has had a significant impact on other techniques such as the Interrogator and the Ina Test tools.

2.1 The Interrogator by Millen, Clark and Freedman

The Interrogator [3] is a Prolog-based software tool that attempts to locate protocol security flaws by an exhaustive search of the state space. In the Interrogator, 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 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. The Interrogator has been successfully used to locate a number of known security weakness [3,7] such as the flaw contained in the Needham-Schroeder key-distribution protocol. An example of this protocol and its flaw is given in figures 1 and 2.

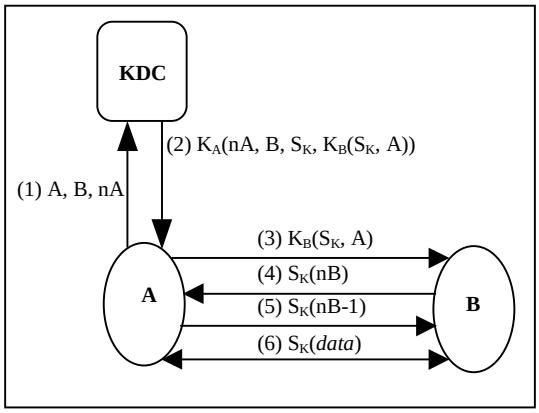

Figure 1: Needham-Schroeder Protocol

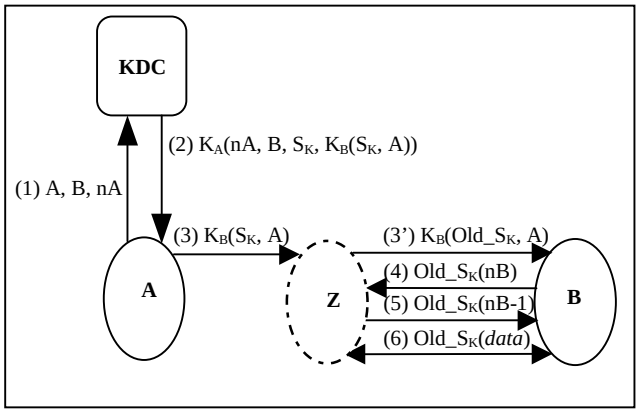

Figure 2: Needham-Schroeder Protocol with flaw

The Interrogator technique first involves representing each participating principle as a finite state machine, as in figure 3.

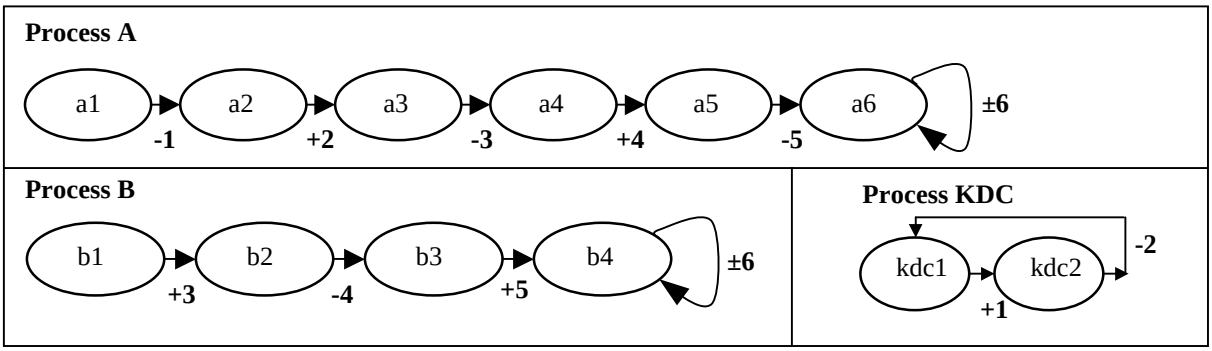

Figure 3: Finite state machine representations for N-S Key Distribution protocol

The transitions between states are specified in terms of *transmit* and *receive* states and are stored in a relations table as seen in Table 1.

The user now provides the Interrogator with a penetration goal. This is some data item which a penetrator might attempt to learn. The Interrogator uses the relations table for the protocol and performs an exhaustive search for any possible penetrations. If a penetration is found, then the history of messages is listed for examination by the user. The Interrogator technique when applied to the Needham-Schroeder key-distribution protocol showed that a weakness exists in step 3, where it is possible to substitute an old previously compromised session key thereby compromising the data sent in step 6.

Process	Relation	Prior State	Message	Next State
a	Transmit	[a,a1]	A→KDC: A,B,nA	[a,a2]
	Receive	[a,a2]	A←KDC: $K_A(nA,B,S_K,K_B(S_K,A))$	[a,a3, $S_K,K_B(S_K,A)$]
	Transmit	[a,a3, $S_K,K_B(S_K,A)$]	A→B: $K_B(S_K,A)$	[a,a4, $S_K,K_B(S_K,A)$]
	Receive	[a,a4, $S_K,K_B(S_K,A)$]	A←B: $S_K(nB)$	[a,a5,S_K, $K_B(S_K,A),nB$]
	Transmit	[a,a5,S_K, $K_B(S_K,A),nB$]	A→B: $S_K(nB-1)$	[a,a6]
	Receive	[a,a6]	A←B: S_K(*data*)	[a,a6]
	Transmit	[a,a6]	A→B: S_K(*data*)	[a,a6]
b	Receive	[b,b1]	B←A: $K_B(S_K,A)$	[b,b2,S_K,A]
	Transmit	[b,b2,S_K,A]	B→A: $S_K(nB)$	[b,b3,S_K,A]
	Receive	[b,b3,S_K,A]	B←A: $S_K(nB-1)$	[b,b4]
	Transmit	[b,b4]	B→A: S_K(*data*)	[b,b4]
	Receive	[b,b4]	B←A: S_K(*data*)	[b,b4]
kdc	Receive	[kdc,kdc1]	KCD←A: A,B,nA	[kdc,kdc2, A,B,nA]
	Transmit	[kdc,kdc2, A,B,nA]	KCD→A: $K_A(nA,B,S_K,K_B(S_K,A))$	[kdc,kdc1]

Table 1: State Relations table for N-S Key Distribution protocol

The Interrogator currently provides the following:

- Analysis of key distribution protocols.
- Deals with conventional key cryptography.
- Searches for security breaches where a penetrator gains knowledge of some secret data.
- Is an automated tool that requires very little input from the user.

2.3 The Ina Jo Technique of Kemmerer

The approach of Kemmerer [4,7] was to utilise an already available machine-aided formal verification system in order to analyse encryption protocols. The system used is based on the Ina Jo specification language [8], which is a nonprocedural assertion language that is an extension of first-order predicate calculus.

The Kemmerer technique defines a cryptographic protocol using finite state machines. One state is differentiated from another by the values of state variables, and the values of these variables can only be changed using well-defined state transitions. The key elements of this approach are types, constants, variables, definitions, initial conditions, criteria, axioms and transforms.

For example: since an Ina Jo Axiom is an expression of a property that is assumed, Kemmerer is assuming that the Encrypt and Decrypt functions are commutative by using the following Ina Jo Axiom in his specifications.

AXIOM $\forall$ *t*:Text, *k*1,*k*2:Key (Encrypt(*k*1, Decrypt(*k*2,*t*)) = Decrypt(*k*2, Encrypt(*k*1,*t*))

This states that for all text *t*, decryption followed by encryption with the matching key pair, or encryption followed by decryption with the matching key pair results in the plaintext message *t*.

The critical requirements that the system is to satisfy in all states are expressed in the criterion clause of the formal specification. An example of a criterion is:

CRITERION $\forall$ k:key (k $\in$ Intruder_Info $\rightarrow$ k $\notin$ Keys_used)

This states that any key that the intruder has access to must not be a key that was used by the system, in the present, past or future.

The transform is a state transition function, which specifies what the values of the state variables will be after the state transition relative to their values before transition.

After the formal specification is completed one can verify the theorems that are generated to check if the critical requirements (Ina Jo criteria) are satisfied. If the theorems are verified and the encryption algorithms satisfy the assumed axioms then the system will satisfy its critical requirements. Kemmerer verifies the system specification using a system test tool called Inatest. This tool allows Ina Jo specifications to be analysed by symbolically executing the formal specifications. Using this tool, the user can input various protocol scenarios, which are then tested by simulation execution. This testing is most useful if a user is unable to verify the Ina Jo theorems and therefore suspects a protocol flaw.

The Inatest based technique is very similar to the Interrogator tool. The major difference between the two tools is that the Interrogator automatically searches scenarios exhaustively for protocol penetrations, whereas the Inatest tool depends on user interaction. One could perceive this interaction as a drawback, as a user could miss a scenario, which could lead to a protocol flaw remaining hidden. However Kemmerer believes that for more complex encryption protocols it will not be feasible to try all paths, and human interaction will be required to make educated decisions.

3. Algebraic Modelling

Another approach to applying formal methods to cryptographic protocol analysis is to model the protocol as an algebraic system, similar to the way in which Dolev and Yao model protocols, but to use algebra to express the state of the participants' (including the intruders) knowledge about the protocol. This is an area that has not received as much attention as the state machine models discussed above. However, given the fact that it is able to combine a detailed model as in the state machine approach with an ability to reason about evolution of knowledge, comparable to that found in logics of knowledge and belief, means that it merits a closer look.

3.1 The protocol histories approach of Toussaint

The protocol analysis technique of Toussaint [9] is based on examining the knowledge states of participating principals in different protocol states. The technique consists of three parts:

- Modelling the protocol specification
- Representing the knowledge of the protocol participants at each stage of the protocol execution
- Analysing the security of the protocol

In order to model a protocol specification, the protocol is first divided into a set of communicating processes, each one known as a participant. The information elements relating to a cryptographic protocol are cryptographic constants and cryptographic variables. A cryptographic protocol is modelled by means of a protocol execution tree. This consists of all global protocol states possible according to the protocol specification. Each protocol state is represented by the cryptographic variables appropriate to that state. In addition to the protocol execution tree, there exists an extended protocol execution tree. This is similar to a protocol execution tree but the possible global states are not restricted to those set out in the protocol specification. It is assumed that any action can be executed by the protocol.

Toussaint argues that since the purpose of a cryptographic protocol is to prevent some principals gaining knowledge of some information, it is natural to base a protocol analysis

technique on the knowledge sets of the protocol participants and intruders. Three different types of knowledge sets are identified:

- The set *F* of fixed elements : the members of this set denote knowledge of the sequence of bits of a message and their meaning (e.g. knowing a cleartext message)
- The set *V* of variables : the members of this set denote knowledge of the existence of a message without knowledge of the sequence of bits representing it (e.g. knowing that a session key exists for use between two principals without knowing that key)
- The set *SV* of semi-variables : the members of this set denote knowledge of the sequence of bits of a message without knowledge of their meaning (e.g. knowing a ciphertext message)

The complete knowledge set *K* is given by $F \cup V \cup SV$

In analysing the security of cryptographic protocols, attacks are considered to arise from two possible sources. Firstly, legitimate protocol participants can attempt to cheat by sending messages that are not allowed in the protocol specification. Secondly, non-legitimate intruders can attempt to intervene at any point in the protocol execution.

An attack on a cryptographic protocol is deemed to be successful if the non-cheating participants are not aware of the presence of the cheating participants or intruders (i.e. the states of knowledge of the non-cheating participants remain consistent with the protocol specification). To incorporate this view of a protocol breach into the analysis technique, Toussaint introduces the concept of a *protocol history*. This is an execution of the protocol for which, although it may include messages which are not allowed in the protocol specification, the non-cheating participants still believe that the execution is in accordance with the specification. Thus, a protocol history is equivalent to a branch of the **extended** protocol execution tree which the non-cheating participants consider to be a possible branch of the **non-extended** protocol execution tree.

It is also possible to have a protocol history favouring a participant suspected of cheating. With such a history, only the participant being favoured is assumed to be aware of all message exchanged during the protocol. All non-cheating participants are assumed to be aware only of the exchanges in which they are directly involved.

Toussaint defines a cryptographic protocol as being secure if every protocol history favouring any participant corresponds to a branch of the non-extended protocol execution tree. In other words, for every execution of the protocol such that all non-cheating participants believe it is in accordance with the protocol specification, if the execution actually is in accordance with the specification, then the protocol is secure. In order to analyse a protocol, it is necessary to successively consider that each participant who can be an intruder is favoured.

A point of interest regarding Toussaint's technique is that the analysis involves a form of scenario testing, as did Kemmerer's Inatest tool. That is, the user must choose a particular protocol history and see if it corresponds to a branch of the non-extended protocol execution tree. As pointed out by Kemmerer, this form of testing suffers from the problem that if the correct scenario is not tested, then the appropriate protocol flaw is not revealed [4]. However, with Kemmerer's method, the user is guided towards particular scenarios by the failure of particular Ina Jo theorems. In Toussaint's case, there is no such guidance and intuitive insight is required in order to detect any protocol flaws. Some form of automated technique (as suggested by Toussaint as the next step) for exhaustive testing would be useful in complementing the analysis. However, this would require some method of reducing the number of search paths which can become infeasible numerous for more complex protocols.

4. Summary

Research in the area of algebraic modelling has not been as active as research in developing and applying state-machine models. However, the success of algebraic modelling techniques in representing very subtle kinds of knowledge[10] and the fact that the objects modelled correspond strongly to entities and messages used in the tools based on state machines suggest that these models could be used to provide the state machine tools with a stronger capability of modelling the knowledge that an intruder could gain. As yet little research has been done on this problem, this leaves the general question of whether and how algebraic modelling can be incorporated in state machine analysis tools.

Table 2 below provides a brief comparison between the different techniques discussed in this paper.

Techniques	Capabilities/Characteristics			
	Category	**Automation**	**Cryptography**	**Application**
Dolev & Yao	Mathematical	No	Public-Key	Secrecy
Millen, Clark & Freedman	State Machine	Yes	Conventional Key	Key Management Protocols
Kemmerer	State Machine	Yes	Conventional and Public-Key	General Purpose
Toussaint	Algebraic	No	Conventional and Public-Key	General Purpose

Table 2: Comparison of Non-Logical Protocol Verification Techniques

5. References

[1] Needham, R.M. and Schroeder, M.D. “Using encryption for authentication in large networks of computers”, Comm. ACM, Vol.21, No.12, 1978, Pages 993-999.

[2] Coffey T. and Saidha P.,”Logic for verifying public-key cryptographic protocols”, IEE Proceedings - Comput. Digit. Tech., Vol. 144, No. 1, January 1997.

[3] Millen J., Clark S.C., and Freedman S.B., “The Interrogator: Protocol Security Analysis”, IEEE Trans. on Software Engineering, SE-13, No.2, 1987, Pages 274-288.

[4] Kemmerer R., “Analysing Encryption Protocols Using Formal Verification Techniques”, IEEE Journal on selected areas in Comms., Vol.7, 1989, Pages 448-457.

[5] Dolev D. and Yao A., “On the Security of Public Key Protocols”, IEEE Transactions on Information Theory, Vol.29, No.2, 1983, Pages 198-208.

[6] Dolev D., Even S., and Karp R., “On the Security of Ping-Pong Protocols”, Information and Control, 1982, Pages 57-68.

[7] Kemmerer R., Meadows C., and Millen J, “Three systems for Cryptographic Protocol Analysis”, Journal of Cryptology, Vol.7, No.2, 1994.

[8] Scheid J. and Holtsberg S. , “Ina Jo Specification Language Reference Manual”. System Development Group, Unisys Corp, Santa Monica, CA. Sept. 1988.

[9] Toussaint, M.J., “A New Method for Analysing the Security of Cryptographic Protocols”, IEEE Journal on selected areas in Comms., Vol.11, No.5, 1993, Pages 702-714.

[10] Meadows C.A., “Formal Verification of Cryptographic Protocols : A Survey”, Advances in Cryptology - AsiaCrypt 1994. Vol. 917, Pages 135-150.