ANALYSIS OF CRYPTOGRAPHIC PROTOCOLS II

Backward State Analysis of Cryptographic Protocols Using Coloured Petri Nets

Darryl M. Stal, Stafford E. Tavares and Henk Meijer†

Department of Electrical and Computer Engineering
†Department of Computing and Information Science
Queen's University
Kingston, Ontario, Canada K7L 3N6
e-mail: tavares@ee.queensu.ca

Abstract — In this paper, we present an approach to analyzing cryptographic protocols using coloured Petri nets. Petri nets provide a graphical specification of a protocol showing the flow and control of information within and between protocol entities. A Petri net model produces a formal description and is used as a basis for evaluating protocols. An explicit intruder model is developed and used to formulate attacks on the legitimate protocol entities. From these attacks, a state diagram is generated. Protocol objectives and modelled assumptions are used to define a set of insecure states. These insecure states are used in the backward state analysis to look for security flaws. We apply our analysis methods to two protocols which use symmetric key cryptography: a published protocol by Hwang [1] and a similar protocol in an ISO working draft [2]. We then verify that the cryptographic protocol does not reach the insecure states defined above. The protocols are also compared and we issue a warning about a possible weakness in the Hwang protocol.

1 Introduction

1.1 Specification and Analysis of Cryptographic Protocols

In this paper, we use Petri nets as a formal method to produce a graphical specification for cryptographic protocols using symmetric key techniques. The Petri net and its associated tools can be used effectively to analyze cryptographic protocols for security weaknesses in the face of an intruder attack. The Petri net methodology is based on a well established tool which can also be used to analyze general protocol properties (e.g., liveness, deadlock, livelock, and boundedness) [3].

There are other formal methods being used to analyze cryptographic protocols. These methods include finite state machines (FSMs) [4], logics [5], and term rewriting [6]. Two of the more recent popular methods are the term rewriting and the logic methods.

The term rewriting model described by Meadows in [6] and [7] to analyze private-key protocols is an adaptation of the public key model. The method is designed for analyzing the security properties of term rewriting systems using algebraic term rewriting operations. In this method, cryptographic protocols are modelled as a set of possible intruder actions. A drawback to this method arises from having to realize all possible intruder actions, in order to make the analysis complete.

In logic methods, such as BAN developed by Burrows, Abadi, and Needham [5], the protocol and its goal are mapped to a set of logical assumptions which are based on knowledge and belief in the protocol. The assumptions are then analyzed using formally defined inference rules to determine whether the goals of the protocol are derivable.

The Petri net method we use is versatile in the sense that it can be used both as a tool to specify protocols and to analyze their security. Furthermore, Petri nets can be used to model all types of distributed systems, not just protocols. While it is generally recognized that logic methods do not give explicit protocol specifications, Petri nets, however, do. An explicit protocol specification can provide warnings of improper use of the protocol, identify remaining ambiguities in the specification, and determine unspecified protocol actions.

Because a Petri net is both a graphical tool as well as a mathematical tool, the benefit of utilizing Petri nets is two fold. Firstly, because the Petri net is a graphical representation of a protocol, it allows one to easily follow the flow of messages within and between protocol entities. Any protocol operation can be modelled by a Petri net diagram.

Secondly, another benefit derives from analyzing the flow of tokens in the Petri net. The state of a Petri net is based upon the distribution of tokens over the system. A Petri net diagram is a very compact representation of a system. In fact, the corresponding FSM has a number of states which can be exponential in the number of places in the Petri net diagram. The system states form the basis for a method of state analysis called backward state analysis. Starting from a known insecure state (e.g., a secret session key is divulged), it is possible to determine, by working backwards, if that state is reachable. If that insecure state is reachable, then a security weakness exists.

A related method for analyzing the state of the system is the use of forward analysis (reachability analysis). In [8] [9] [10], a forward reachability analysis of the Petri net model produced a full reachability tree containing all reachable states within the system. An exhaustive search through the tree was performed to determine if an insecure state could be reached. For a large system, the analysis suffers from state explosion, where the number of states increases exponentially. One can "prune" this tree by applying the acceptance criteria of the protocol entities to the intruder. Hence, the intruder can only generate messages which will be accepted by the protocol entities. This technique leaves only those branches that result in a successful completion of the protocol [10]. One of our goals is to automate the verification of cryptographic protocols modelled using coloured Petri nets.

1.2 Petri Net Basics

Petri nets, first described by Carl Petri in [11], are a useful tool for specifying and analyzing a variety of distributed systems, including cryptographic protocols [8] [9] [10] [12]. An ordinary Petri net is a directed graph with two kinds of nodes: places and transitions, connected by directed arcs. A place, drawn as a circle, represents a condition in the system, and a transition, drawn as a rectangle, represents the occurrence of an event in the system. Tokens, drawn as black dots, occupy places to represent the truth of the associated condition. The distribution of tokens over the Petri net is called a marking, which represents the state of the system.

A transition is enabled and can fire when each of its input places contains at least one token. When a transition fires, an event has occurred within the system, and a change in the system state

occurs. **Figure 1** shows an ordinary Petri net and its change of state resulting from the firing of transition T_2. According to the transition firing rules, one token is removed from each input place and one token is added to each output place.

There are three other conventions which require explanation. Firstly, the doubled-headed arrow between P_2 and T_2 means that place P_2 is acting both as an input place and an output place so a token will be removed as transition T_2 fires but then replaced. Secondly, the small circle attached to T_1 changes the arc between T_1 and P_1. That arc is now a composite of an ordinary arc and an inhibitor arc. This means that transition T_1 cannot fire again until the token in P_1 has been cleared by transition T_2, as shown in **Figure 1 (b)**. Lastly, transition T_1 represents a "source" transition, meaning it can generate tokens at will.

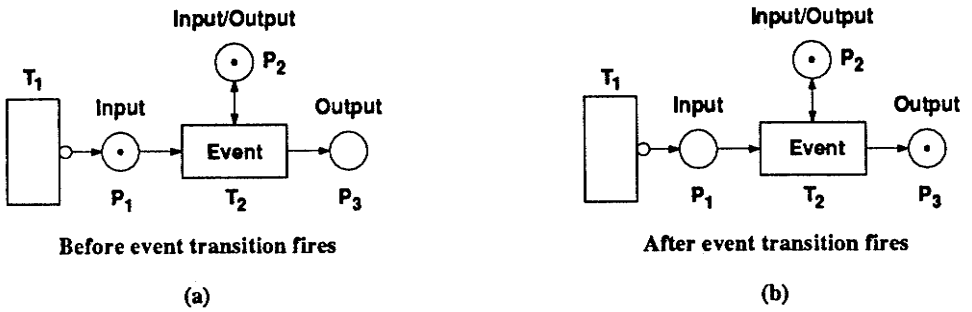

Figure 1 Transition firing rule for Petri nets.

As an extension of ordinary Petri nets, coloured Petri nets have been developed to produce more compact and manageable descriptions. In a coloured Petri net, a place can host tokens with different meanings (i.e., colours). Each place and transition has an attached set of colours. When a transition fires, tokens are added and/or removed from places, as in ordinary Petri nets. However, there is a functional dependency between the colour of the transition which is firing, and the colour of the tokens involved. Hence, only a certain colour (or set of colours) will enable any given transition. The colour of a token often represents a complex data value, and can change with the firing of a transition. For these reasons, coloured Petri nets are suited for a wide range of distributed systems [13]. For more information on ordinary and coloured Petri nets, see [3] [13] [14] [15].

1.3 Intruder and Protocol Entity Modelling

A cryptographic protocol should be designed to combat attacks from an intruder who has control of the communications medium and tries to insert, modify, delay, or delete partial or complete contents of the messages transferred over the channels. To verify whether a protocol actually has the desired properties that can protect the system against intruder attacks, we need to model these attacks and test the performance of the protocol under these attacks.

A coloured Petri net model of a cryptographic protocol is composed of protocol entity modules and possibly an intruder. The protocol entity modules are derived from the protocol descriptions detailing the information flow between and within protocol entities, as first described in [8]. Large rectangles called modified or "super" transitions, represent protocol entities. They are connected

to the rest of the system through small boundary transitions called ports. A high-level or "black box" Petri net diagram is shown in **Figure 2.**

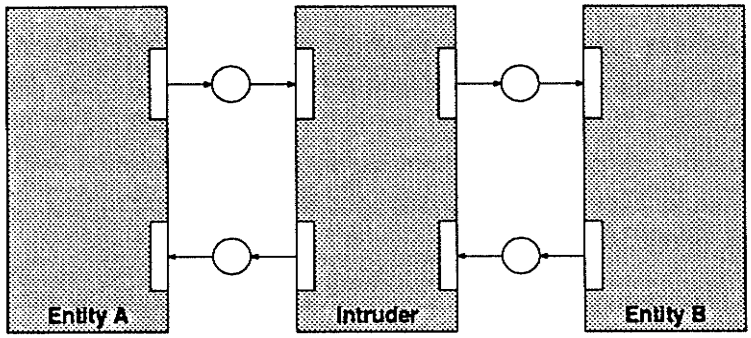

Figure 2 High-level or "black-box" view of intruder attacking protocol entities.

The intruder model is composed of dedicated intruder processes, each designed for a specific message in the protocol. Each intruder process has access to the intruder's information resources during an attack. The information resources contain extracted information from intercepted messages and from the protocol specification. Each intruder process has the ability to intercept a message from the channel, delete it, return an altered version back to the channel, or synthesize a spurious message and inject it into the channel. We assume that the intruder has complete knowledge of the protocol, a record of previous transactions between the protocol entities, and the ability to perform any operation defined in the protocol. We further assume that the intruder has no direct access to secret information such as secret keys stored within a protocol entity.

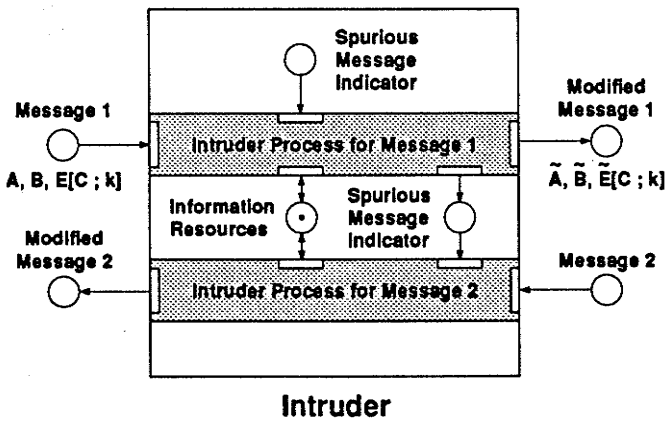

Figure 3 Low-level view of intruder model, with information flow.

As first described in [10] and [12], a low-level intruder model with an example information flow is shown in **Figure 3.** In this example, the intruder receives the input message {**A, B,** E[**C ; k**]} where **A** and **B** are messages sent in the clear and E[**C ; k**] represents a message, **C** encrypted

under the key, **k**. Since the message may be manipulated by the intruder, the output message will be shown as [$\tilde{\mathbf{A}}$, $\tilde{\mathbf{B}}$, $\tilde{\mathbf{E}}$[**C** ; **k**]] to indicate that each variable in the message may have been modified. The symbol tilde (˜) indicates that the variable could be either modified (circumflex, ˆ) or unmodified (no accent on variable).

1.4 Backward State Analysis

Our method of modelling and verifying cryptographic protocols with coloured Petri nets and backward state analysis appears to fall in both the *Type I* and *Type II* classifications given by Meadows in [6].

The *Type I* classification comprises a group of techniques that attempt to model and verify a protocol using specification languages and verification tools not specifically designed for the analysis of cryptographic protocols. Coloured Petri nets can be considered a specification language because a coloured Petri net model gives an explicit protocol specification. Furthermore, Petri nets were not originally developed for the analysis of cryptographic protocols [11]. Hence, coloured Petri nets meet the criteria for this classification.

The *Type II* classification comprises a group of analysis techniques that does not guarantee that the investigation of a protocol's security is complete. These techniques, however, can be used successfully to identify either known or previously unknown flaws in a protocol. Our backward state analysis technique seems to meet the criteria for this classification.

Backward state analysis of a cryptographic protocol involves two steps:

1. identifying insecure states that may or may not occur and
2. performing a separate backward state analysis for each insecure state to test if each particular insecure state could exist or not.

The insecure states could be any action that constitutes an unintended or improper use of the protocol (e.g. a secret key is divulged to an intruder, a legitimate user accepts an invalid message as correct, etc.). When the insecure state is identified, we must look exhaustively for a state path from that final insecure state to a valid initial state. If a path exists, given our intruder model, then the insecure state is reachable and hence, that insecure state could occur. If no path exists, then the insecure state is unreachable and hence, that insecure state could not occur. Other possible insecure states must be verified individually through the same backward state analysis technique. That means performing an exhaustive state search from the chosen insecure state to a valid initial state. Although this method of analysis may not identify all insecure states, from our experience, this method may prove useful in identifying security weaknesses in cryptographic protocols.

2 Hwang's Scheme for Secure Digital Mobile Communications

Although Petri net analysis can be applied to different types of cryptographic protocols, we are most interested in mobile communications because of the potential for fraudulent usage and our interest in secure wireless communications. Our work will attempt to verify the correctness of the Hwang protocol [1]. This protocol was chosen as an example for our method of specification and analysis for two reasons: it is designed for use in a digital mobile communication system and its structure is similar to an ISO working draft protocol [2].

The purpose of this protocol is to efficiently distribute a secret key for any two terminals in the system in such a way that any user can log in to any terminal and communicate with any other user at another terminal in a secure way. From the coloured Petri net model of the Hwang protocol, given in **Figure 4**, a protocol run is initiated when a user (User **A**) wishes to have secure communications with a second user (User **B**). User **A** initiates a call by sending the message $\{\mathbf{n_1}, \mathbf{ID_a}, \mathbf{ID_b}\}$ to User **B** from the Send Request output port. User **B** then adds the nonce $\mathbf{n_2}$ to this message and forwards the modified message to the Server **S**. The Server then generates a session key, SK, for the two parties to share. Both users get a copy of the session key, encrypted under their own individual secret key (K_a and K_b). Both users will receive these encrypted messages and will decrypt them using the secret keys contained in the IC cards.

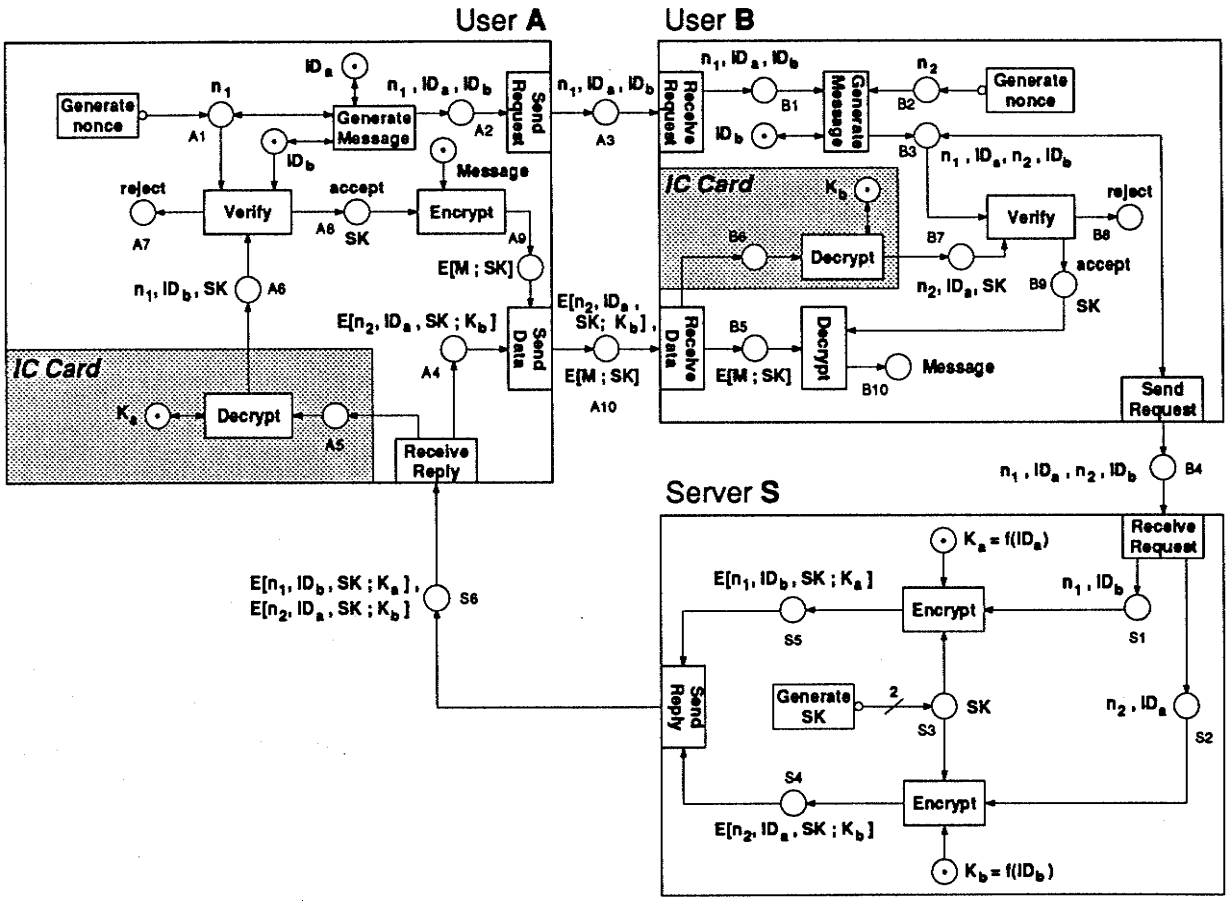

Figure 4 Petri net model of the Hwang protocol.

For all coloured Petri net diagrams, all places holding non-permanent data are designated with a letter-number pair (e.g. A3, B1, C4). This numbering scheme is also used to define the states in the backward state analysis. It should be pointed out that in **Figure 4**, the **2** located between the Generate SK transition and place S3 (in the Server entity) implies that the transition generates two tokens and places both in place S3.

2.1 Analysis of the Hwang Protocol

After applying intruder attacks on this protocol, whereby an intruder entity is placed between legitimate protocol entities, our analysis found a weakness in the authentication stage of the protocol. An intruder was introduced between legitimate protocol entities to intercept and otherwise tamper with transmitted messages. A situation was discovered whereby User A may accept invalid messages as being correct. This came as a result of investigating the possible outcomes of altering one or both of the plaintext messages in Message 1 ($A3=\{n_1, ID_a, ID_b\}$) and Message 2 ($B4=\{n_1, ID_a, n_2, ID_b\}$).

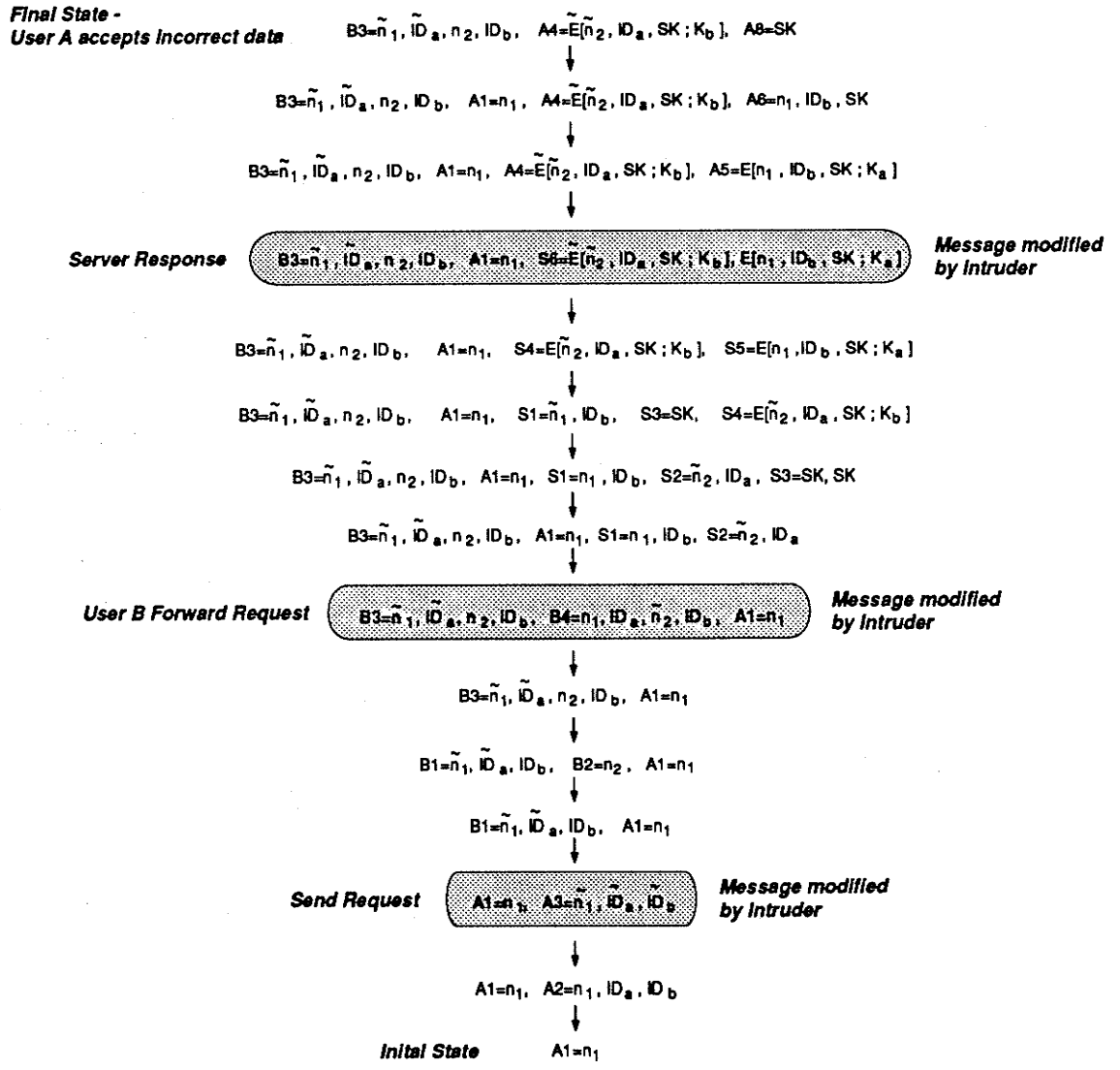

Figure 5 Backward state analysis results from attack on Hwang protocol.

Hence, in the backward state analysis diagram in **Figure 5**, we have identified as an insecure state, the acceptance of an invalid message in User A. From this final insecure state, a backward

state analysis was carried out. Working backwards through the previous states, a full state tree was developed from the final insecure state to a valid initial state. Hence, the insecure state was reachable and could occur. Next, a modification symbol (~) was placed over every variable in the top line of the backward state analysis. This was to determine the maximum extent of modified messages that would still result in User **A** accepting an invalid message.

Because the protocol could fail if certain messages were modified, we applied the acceptance criteria of the protocol entities to the intruder to determine which messages could be modified and which could not. The acceptance criteria forces the intruder to only generate messages which will be accepted by the protocol entities. This has the effect of removing the modification symbol from certain variables and messages. Hence, the results given in **Figure 5** represent the full extent of modifications that would still lead to the insecure state described above.

Other insecure states have been identified as candidates for backward state analysis. For example, a situation may arise whereby User **A** believes it is authenticating User **B** but in fact is authenticating another user altogether. This fraud may be accomplished by modifying **Message 1** (**A3**=$\{n_1, ID_a, ID_b\}$) and **Message 2** (**B4**=$\{n_1, ID_a, n_2, ID_b\}$). By changing ID_b to ID_c in **Message** 1 and ID_c back to ID_b in **Message 2**, User **A** will believe it is authenticating User **B**, but in fact is authenticating User **C**. Although User **C** has been improperly authenticated, it will not be able to read any messages because it still lacks User **B**'s secret key, K_b. The reachability of this insecure state was tested for and verified using a separate backward state analysis tree.

The protocol was also analyzed, using a separate backward state analysis tree, to see whether User **B** could be deceived in a similar way as User **A**. An intruder or legitimate protocol entity could impersonate User **A** and send to User **B** an incorrect source ID. User **B** would have no way of knowing the true identity of the sender. The protocol would then proceed normally. At the end of the protocol run, User **B** would believe it had correctly authenticated the user that matched the original source ID sent to it. In reality, it would have authenticated an untrustworthy user or an outside intruder.

2.2 Security Equivalence in Hwang's Protocol

Security equivalence refers to a property exhibited by certain protocols that share the same level of security. In the case of Hwang's protocol, certain modifications can be made without altering the level of security provided. Specifically, by changing the path of some data, the result is a more conventional information flow without a loss in security. By changing the path of the second half of the message in S6 (i.e. $E[n_2, ID_a, SK ; K_b]$) to be directed to User **B** instead of being routed through User **A**, a more natural and efficient protocol results, without an adverse affect on security. Furthermore, this modification makes Hwang's protocol even more similar to the ISO working draft protocol.

3 An ISO Working Draft Protocol

The ISO has proposed several mechanisms in [2] for key management using symmetric techniques. The mechanism we have chosen to include here is *Key Establishment Mechanism 8* — one of the twelve mechanisms proposed for standardization from the above document. The

Petri net model of the ISO working draft protocol is shown in **Figure 6.** It was chosen over the other mechanisms because of its similarity to the Hwang protocol.

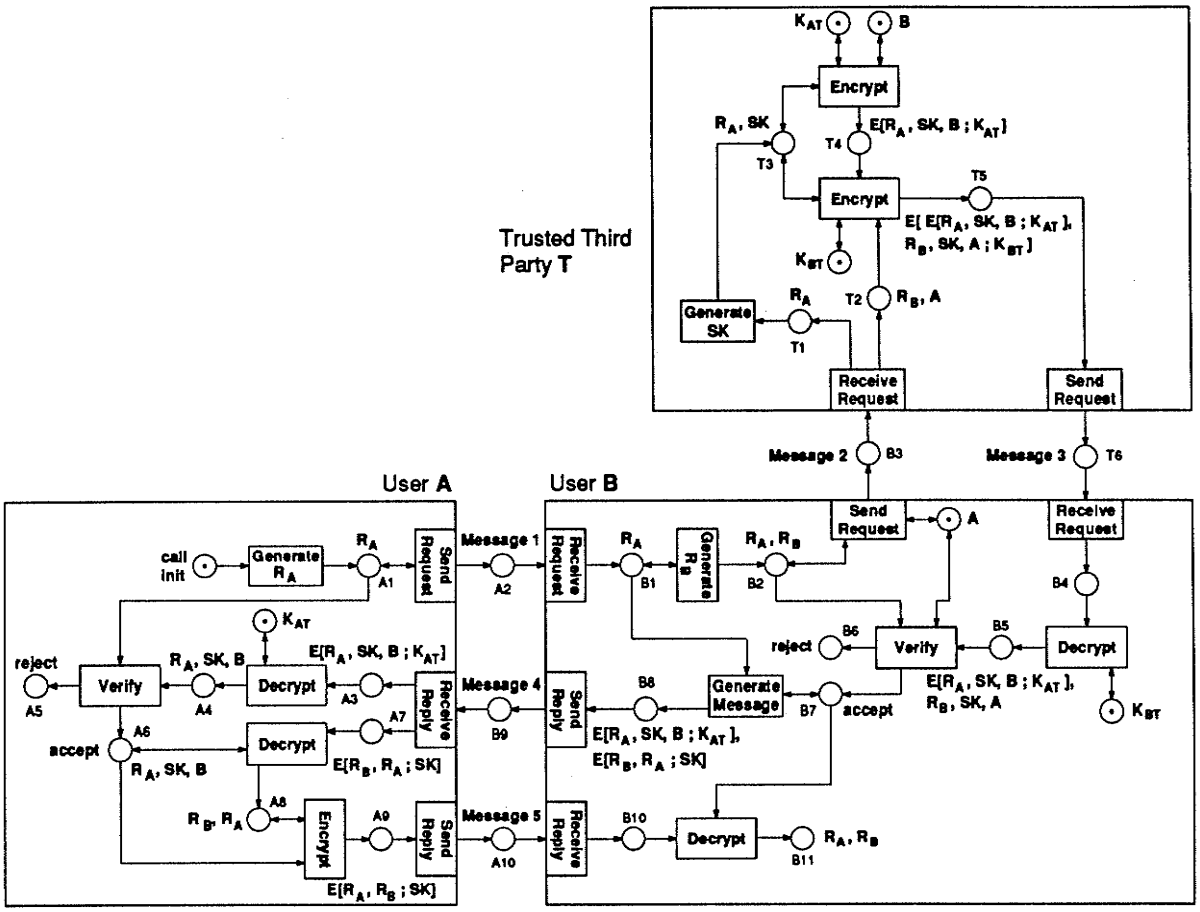

Figure 6 Petri net model of ISO working draft protocol.

Aside from the presence of a third party, there are many other similarities. For example, both protocols have one user initiate the protocol run by contacting the desired party (e.g. User **A** calls User **B**); both protocols use the third party to generate the session key, SK; the third party shares a different secret key with both entities in both protocols; both protocols attempt to provide mutual authentication; and both protocols prevent replay attack through the use of random numbers or nonces. Furthermore, if **Message 3** from the ISO working draft protocol were directed to User **A** instead of User **B**, and **Message 4** was removed, the path of the message flow would be identical in both the ISO working draft protocol and Hwang's protocol.

In this mechanism, User **A** initiates a call by sending a request to User **B** from the Send Request output port. User B then sends a message to the third party, which generates a session key for both users to share for the duration of their call. The session key is then distributed to both users through User **B**. The messages containing the session keys, **Messages 3** and **4**, are verified upon receipt by Users **B** and **A**, respectively.

3.1 Analysis of the ISO Working Draft Proposal

An intruder attack was applied to this protocol. The results of an attempted attack are shown in the backward state analysis in **Figure 7**. The failed attack results are given below because we could not find a flaw/weakness in our analysis of this protocol.

To perform the backward state analysis on this protocol, a final possible insecure state was identified to show that this state could not be reached, and, hence, that no security flaws could be found. In the analysis shown, we chose the final state (**I7**={E[$\mathbf{R_B}$, **SK, B** ; $\mathbf{K_{AT}}$], $\mathbf{R_B}$, **SK, A**])) to be the intruder learning the session key, SK. This was accomplished by having the intruder attempting to impersonate User **B** at the reception of **Message 3** from the Third Party, **T**. The attack failed because the Intruder, **I**, could not successfully impersonate User **B** or any of the legitimate protocol entities due to a mutual authentication mechanism designed to prevent impersonation.

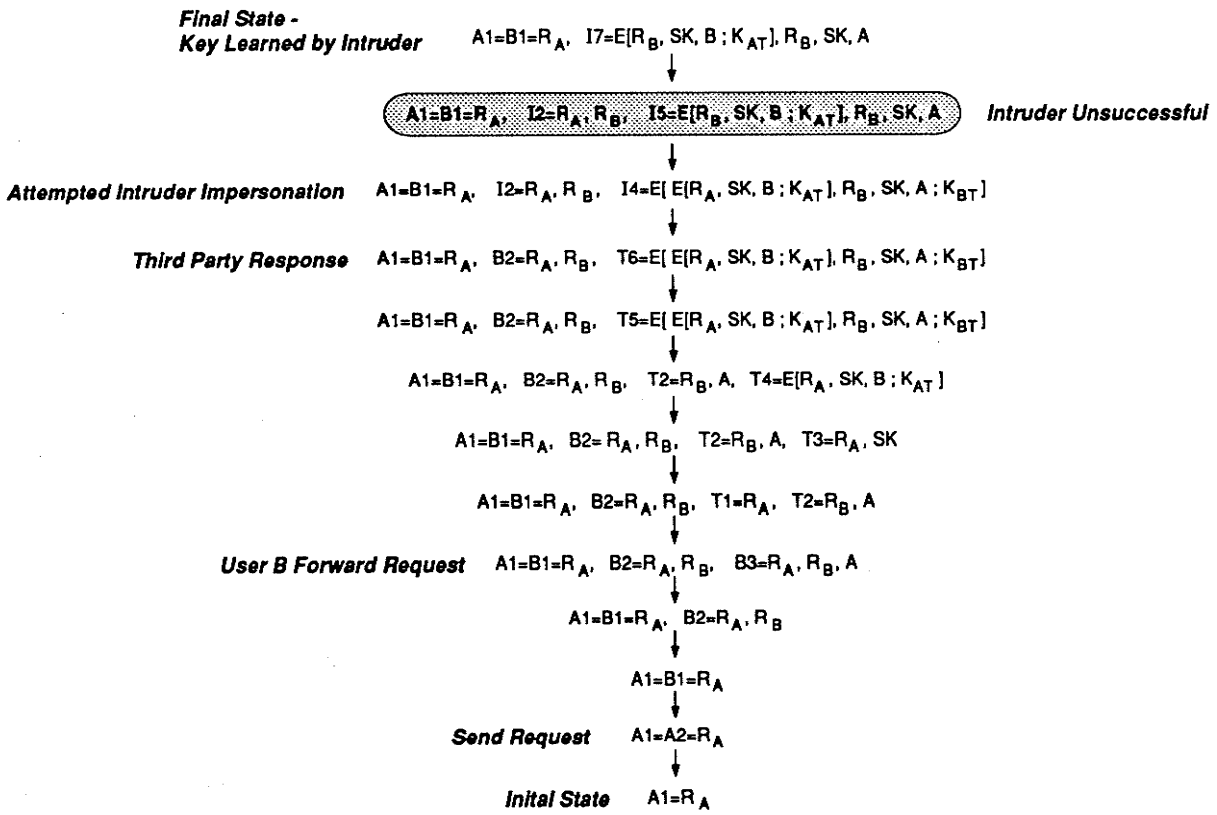

Figure 7 Backward state analysis results from attack on ISO working draft protocol.

Apart from not being able to learn the session key, SK, the intruder is not able to learn the secret keys, K_{AT} or K_{BT}, for the reason that they are kept secret. Furthermore, the protocol uses random numbers to prevent replay attacks. However, an intruder, using its given capabilities from our model, can still disrupt communications for the legitimate protocol entities.

4 Conclusions

The protocol specification and analysis method presented in this paper is capable of providing a graphical specification of cryptographic protocols and their threats. Used in conjunction with the backward state analysis technique, our method allows a protocol to be analyzed for potential security weaknesses.

Two protocols were chosen for analysis: the Hwang protocol and an ISO working draft protocol. These protocols were chosen, firstly, for their application to mobile communications and secondly, for their similarity to each other.

A Petri net model was developed for both the Hwang protocol and the ISO working draft protocol, describing the information flow within and between protocol entities. An intruder model was placed between all protocol entities. A series of system states were derived from the Petri net diagram. Backward state analysis was carried out for both protocols.

The backward state analysis technique detected insecure states in the Hwang protocol. One insecure state permits legitimate protocol entities to accept incorrect information. This could lead to the incorrect authentication of either protocol entity. Two other distinct insecure states were identified and verified. Furthermore, an equivalent and more efficient model was suggested for the original Hwang protocol, based on security equivalence criteria. This modification would give the protocol a more conventional information flow. Our analysis of the ISO protocol found no security flaws or weaknesses. That is, based on our intruder model, no paths could be found from an insecure state to a valid initial state.

5 Status of the Research Project

This paper reports on ongoing research into the application of Petri nets and backward state analysis to the specification and verification of cryptographic protocols. Although more work needs to be done, we have not yet discovered any fundamental limits to the method.

Some of the aspects that we are currently, or planning on, pursuing include:

- investigating automation of the backward state analysis technique,
- looking at systematic methods for identifying larger classes of insecure states,
- introducing a database into the intruder entity to analyze the effect of this capability for attacks on multiple iterations of a protocol, and
- studying the complete forward analysis of a protocol to show that, given the intruder model, no insecure states that have been identified can be reached.

6 Acknowledgment

The authors would like to thank Paul van Oorschot for his remarks which helped to improve this paper.

Bibliography

[1] T. Hwang, "Scheme for secure digital mobile communications based on symmetric key cryptography," *Information Processing Letters*, vol. 48, pp. 35–37, 1993.

[2] ISO/IEC JTC1/SC27/N832. ISO/IEC 11770–2: Information technology — Security techniques — Key Management — Part 2: Key Management Mechanisms Using Symmetric Techniques. Draft Document, International Organization for Standardization, 1993.

[3] R. David and H. Alla, *Petri Nets & Grafcet*. Prentice Hall Inc., 1992.

[4] V. Varadharajan, "Verification of network security protocols," *Computers & Security*, pp. 693–708, 1989.

[5] M. Burrows, M. Abadi, and R. Needham, "A logic of authentication," *ACM Transactions on Computer Systems*, vol. 8, no. 1, pp. 18–36, 1990.

[6] C. Meadows, "Applying formal methods to the analysis of a key management protocol," *Journal of Computer Security*, vol. 1, no. 1, pp. 5–35, 1992.

[7] C. Meadows, "Using narrowing in the analysis of key management protocols," in *1992 IEEE Computer Society Symposium on Research in Security and Privacy*, (Oakland, California), pp. 138–147, May 1992.

[8] B. B. Nieh and S. E. Tavares, "Modelling and analyzing cryptographic protocols using petri nets," in *Advances in Cryptology, Proc. of AUSCRYPT'92*, pp. 275–295, Springer-Verlag, Berlin, 1993.

[9] B. B. Nieh, "Modelling and analysis of cryptographic protocols using petri nets," Master's thesis, Department of Electrical Engineering, Queen's University, Kingston, Ontario, 1992.

[10] C. M. Morton, "A modular approach to modelling cryptographic protocols using petri nets," Master's thesis, Department of Electrical Engineering, Queen's University, Kingston, Ontario, 1993.

[11] C. Petri, *Kommunikation mit Automaten*. PhD thesis, Instiut fur Instrumentelle Mathematik, Schriffen des IIM, 1962.

[12] L. Robart, "Decomposition techniques for cryptographic protocols in wireless communication systems," Master's thesis, Department of Electrical Engineering, Queen's University, Kingston, Ontario, 1993.

[13] K. Jensen, "Coloured petri nets," in *Advances in Petri Nets 1986*, number 254 in Lecture Notes in Computer Science, pp. 248–299, Springer-Verlag, Berlin, 1986.

[14] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[15] M. Diaz, "Modeling and analysis of communication and cooperation protocols using petri net based models," *Computer Networks*, vol. 6, no. 6, pp. 419–441, 1982.

On Unifying Some Cryptographic Protocol Logics

Paul F. Syverson* Paul C. van Oorschot†

4 April 1994

Abstract

We present a logic for analyzing cryptographic protocols. This logic encompasses a unification of four of its predecessors in the BAN family of logics, namely those given by Gong, Needham, and Yahalom (1990), Abadi and Tuttle (1991), van Oorschot (1993), and BAN itself, i.e. Burrows, Abadi, and Needham (1989). We also present a model-theoretic semantics with respect to which the logic is sound. The logic herein captures all of the desirable features of its predecessors and more; nonetheless, it accomplishes this with no more axioms or rules than the simplest of its predecessors.

Keywords: Analysis of Authentication Protocols, Logics of Belief, Cryptographic Protocols.

This paper will be presented at the 1994 IEEE Symposium on Research in Security and Privacy, May 16-18 1994, Oakland, California. It will appear in the proceedings of that conference, published by IEEE Computer Society Press. Copies are available from the authors.

References

[1] Martín Abadi and Mark Tuttle, "A Semantics for a Logic of Authentication", *Proc. of the Tenth ACM Symp. on Principles of Distributed Computing*, pp.201–216, ACM Press, August 1991.

[2] Michael Burrows, Martín Abadi and Roger Needham, "A Logic of Authentication", Digital Systems Research Center, Research Report 39, revised Feb. 22 1989. See also *ACM Transactions on Computer Systems* **8**(1) pp.18–36, Feb. 1990.

[3] Li Gong, Roger Needham and Raphael Yahalom, "Reasoning about Belief in Cryptographic Protocols", *Proc. 1990 IEEE Symp. on Research in Security & Privacy*, pp.234–248, IEEE Computer Society Press, 1990.

[4] Paul C. van Oorschot, "Extending Cryptographic Logics of Belief to Key Agreement Protocols", *Proc. First ACM Conference on Computer and Communications Security*, pp.232–243, Nov. 1993.

*Code 5543, Naval Research Laboratory, Washington, DC 20375 (syverson@itd.nrl.navy.mil)

†Bell-Northern Research, P.O. Box 3511 Station C, Ottawa, Canada K1Y 4H7 (paulv@bnr.ca). Also with: School of Computer Science, Carleton University, Ottawa, Canada K1S 5B6.