Automated Security Analysis of Cryptographic Protocols Using Coloured Petri Net Specifications

(Extended Abstract)

Eric Doyle†, 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: doyle@ee.queensu.ca,
tavares@ee.queensu.ca, meijer@qucis.ee.queensu.ca

Abstract. This paper extends the Petri net methodology for the specification and analysis of cryptographic protocols. In particular, software to perform automated state-reachability analysis has been developed. The intruder and each legitimate protocol entity is modeled by a Petri Net Object (PNO). The goal of the analysis is to determine whether a cryptographic protocol can withstand the attacks of the specified intruder. Hand analysis is not practical in most cases because of the large number of actions the intruder may pursue. However, exhaustive state analysis on a modern workstation is feasible for many protocols. Previous work is extended to model multiple iteration and parallel session attacks. In particular, our approach is able to show that a reflection attack can be executed on a simple one-way authentication protocol. The prevention of the reflection attack is demonstrated through a slight modification of the protocol. The handset authentication protocol used in CT2 and CT2Plus is also modeled and analyzed. A common problem for two-party cryptographic protocols, which is solved by subsequent steps in this case, is confirmed by this analysis. Future work on the methodology is described, and migration of the software from Prolog to an object-oriented programming language such as C++ is planned.

1 Introduction

1.1 Analysis of Cryptographic Protocols

Specification and analysis are two separate, but equally critical, tasks in the development of cryptographic protocols for use in a desired application. The processes executed by the legitimate participants and the messages flowing between them are defined explicitly in the specification. It must be ensured that it is a complete and accurate description of the final implementation. Analysis establishes the correctness of the protocol. It follows that the model used for analysis must match the specification exactly.

As with any communication protocol, verification of structurally dependent properties such as liveness and boundedness must be performed. However, in addition, the analysis must consider the security properties of the cryptographic protocol. An intruder is an additional entity which may block, change, or inject messages on the communication channel between the legitimate parties. Analysis may either show that such attacks succeed in subverting the original goal of the protocol, or prove that they do not. Three approaches to analyzing the security properties of cryptographic protocols are: logical, algebraic, and state-transition-based. Verification methodologies employ one or more of these approaches.

In the logical approach, a protocol is transformed into logical assertions. The logic is a mechanism used to analyze the beliefs of the trustworthy participants in the protocol which evolve as a consequence of the intercommunication. The idealization step required to generate the assertions means that this approach may not be appropriate for defining a complete protocol specification. BAN logic discussed in [1,2] is an example of cryptographic protocol verification based on logic. By revealing previously undiscovered protocol flaws, BAN logic has demonstrated that it is a useful methodology.

A comparison of three algebraic/state-transition-based analysis systems is given in [3]. Kemmerer uses a general-purpose tool for software specification called Inatest. The NRL Analyzer and the Interogator, from Meadows and Millen, respectively, are systems based on Prolog which are specifically developed to analyze cryptographic protocols. In algebraic analysis, expressions representing messages are manipulated according to a defined set of rules. From this it may be determined whether the design goal for the protocol can be violated. State-transition analysis operates on a state model of the protocol. The model is executed to determine whether insecure states can be reached. Flavours of both approaches are offered in each of the three methodologies presented by Kemmerer, Meadows, and Millen. Of the three, the Interogator is the most automated, but each tool has its particular strengths.

1.2 Petri Nets for Specification and Analysis

In the Petri net-based approach used in this paper, the same model is used for both specification and analysis. The method is best described as state-transition-based. Coloured Petri nets [7] form the formal specification language, and have a simple graphical representation (see Figure 1). A Petri net is a graph with two types of nodes: places and transitions. Places are drawn as circles, and may contain any number of dots, called tokens. In coloured Petri nets, each token is assigned a colour which identifies its value. The contents of a place may be restricted to a particular

class of colours, much like the way types restrict the values of variables in high-level programming languages. The distribution of tokens in the set of places determines the state of the system.

The transitions, each depicted by a rectangle containing a label, define how the token distribution will change as the system execution progresses; i.e., they represent events. Each transition has zero or more input and output places. A directed arc is drawn from an input place to the transition, and one is drawn from the transition to an output place. A double-headed arrow indicates that the place is both an input and an output for the connected transition. In a given state, a transition is enabled if appropriate tokens reside in each of its input places (as defined by the transition). Enabled transitions may fire singly by removing a token from each input place (one for each arc), and placing a token in each output place. A table, or a function, defines the relationship between the input values and the output values. Firing changes the system state.

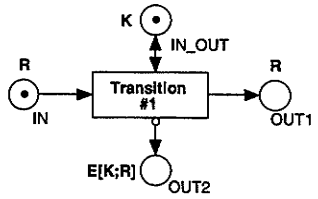

Figure 1. A simple Petri net with one transition, and four places.

An inhibitor arc, a special arc drawn with a small hollow circle on its tail, prevents a transition from firing until the corresponding place is empty. Output/inhibitor arcs, arcs directed to an output place with the inhibitor characteristics, are used in the protocol models. Another extension to Petri nets, which evolved from work described in [8,9], is the Petri Net Object (PNO) which is shown later in Figure 2. The PNO is a Petri net enclosed in a box with transitions at the perimeter called ports. Ports are the only external access nodes in a PNO. The PNO is the "black box" element from which the protocol entities are constructed. A port on a protocol entity is connected to the port on another through an intermediate place which represents the channel. The level of detail shown in the Petri net may be controlled by judicious use of PNOs. While the specification contains implementation-level detail, the Petri net will depict the appropriate level of abstraction. Thus, the Petri net model is a hierarchical tool well suited for iterative analysis and design.

2 Simple One-Way Authentication Protocol

The simple one-way authentication protocol is used by an entity A to verify the presence and identity of another entity B. This protocol employs a challenge-response scheme. The entities A

and B share a secret key k. Entity A sends a random value r (the challenge) to entity B; B responds to A by returning e(r,k), the encrypted value of r under key k, where e is a symmetric encryption function. The returned value is decrypted with key k by entity A. If the result matches the original random value r, A accepts and the protocol terminates with success.

The objective of this protocol is to allow entity A establish that entity B was present (i.e., participated in the message exchange of the protocol) sometime during the interval between sending a challenge and receiving a response.

One reason this protocol is chosen for study is to extend the classes of attacks modeled by the Petri net methodology. It was taken deliberately since it is simple, it exhibits a weakness to reflection attack, and there is a well-known modification which can prevent the attack. Authentication is a recurring objective in many cryptographic protocols, particularly in those designed for wireless communication systems. Studying this simple protocol leads to the specification and analysis of practical, standard protocols such as CT2Plus.

2.1 Petri Net Model of the Protocol

The initial model of the simple one-way authentication protocol is shown in Figure 2. It consists of two Petri Net Objects: one for entity A and another for entity B. These are inter-connected by arcs attached to places which represent the channels through which entities A and B exchange messages. The place/transition structure in each PNO represents the simple sequential process executed by the entity to fulfill its role in the protocol. A single token in each place at the top of each PNO (i.e., places A1 and B6) indicates that each process will execute once during the entire protocol.

The transitions in the PNO represent particular functions which the process may perform while it executes. In this protocol the relationships are simple. For example, transition SEND CHALLENGE consumes a plain (i.e., without colour) token from place A1. It deposits a token with colour r in places A2 and A3. Transition RECEIVE RESPONSE has a more complex definition: take a token with colour X (where X may be any colour) from A3; take a token from B8 and encrypt its value using the colour of the token in A0 as the key; if the result matches the colour X, deposit an X-value token in A5 (accept); otherwise, deposit an X-value token in A4 (reject).

As illustrated in Figure 3, the model of entity A is extended to accommodate a second process (A-2) which executes in parallel with the first (A-1); both share a common key, k. This new process is identical to the process in entity B. It models a process in entity A which is set to respond to an authentication challenge from any other entity. With this approach it is possible to

add an arbitrary number of concurrent processes to each participant. Thus, even multiple-iteration or parallel session attacks [13] may be analyzed using models constructed in this fashion.

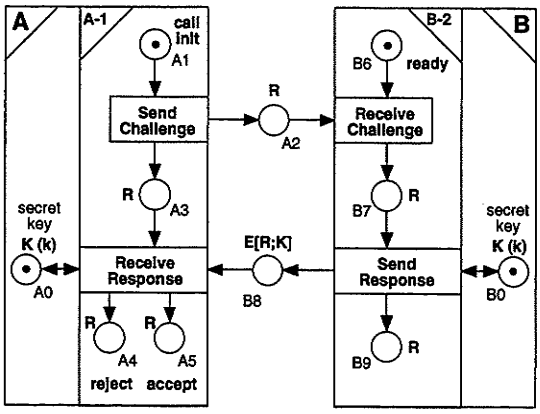

Figure 2. Petri net model of the one-way authentication protocol.

2.2 Petri Net Model of the Intruder

In a cryptographic protocol, an intruder is an entity with some degree of control of the traffic on the communication channels, and which attempts to subvert the objectives of the protocol. Such attempts are termed attacks on the protocol. In the case of the one-way authentication protocol, a successful attack would mean convincing entity A (process A-1) that entity B was present during the protocol execution when entity B never actually participated.

The intruder intercepts and stores all messages delivered to any channel. It may redirect, pass, or block any of these messages. In addition, it may construct spurious messages from captured and randomly generated values, and inject them into any channel.

In the Petri net model of the protocol, the intruder is represented by a PNO. The PNO has ports which mirror the functions of the ports of the legitimate participants. In Figure 3, the port SEND B CHALLENGE mirrors the function of port SEND CHALLENGE in entity A. The database in which the intruder stores all captured and randomly generated values is represented by a set of places, one place for each type of value, e.g., one place to store keys, and another to store encrypted values. The values are stored permanently in the database. Therefore, only input arcs or bi-directional arcs may be connected to these places. The remaining internals of the intruder PNO consist of a network of directed arcs (and possibly extra transitions and places) which define the relationships between the captured messages, the database, and the output messages.

A few rules apply to the construction of the intruder model. An intruder port which captures messages from a channel (input port) will have an arc directed to a place in the database. For example, when the transition RECEIVE A CHALLENGE in Figure 3 fires, it removes a token from place A2 and outputs a token to place I_R in the database.

An intruder port which injects messages into a channel (output port) will be connected to a place in the database by a bi-directional arc. The same port is always connected to the channel place via an inhibitor arc. The implication is that the model of the intruder will deliver messages singly to the channel. The target entity must first remove a sent message from the channel before the intruder injects another. In Figure 3, the transition SEND B CHALLENGE may fire only if the place I4 is void of tokens. When firing, it checks, and returns, a token from place I_R, and outputs a token to the place I4.

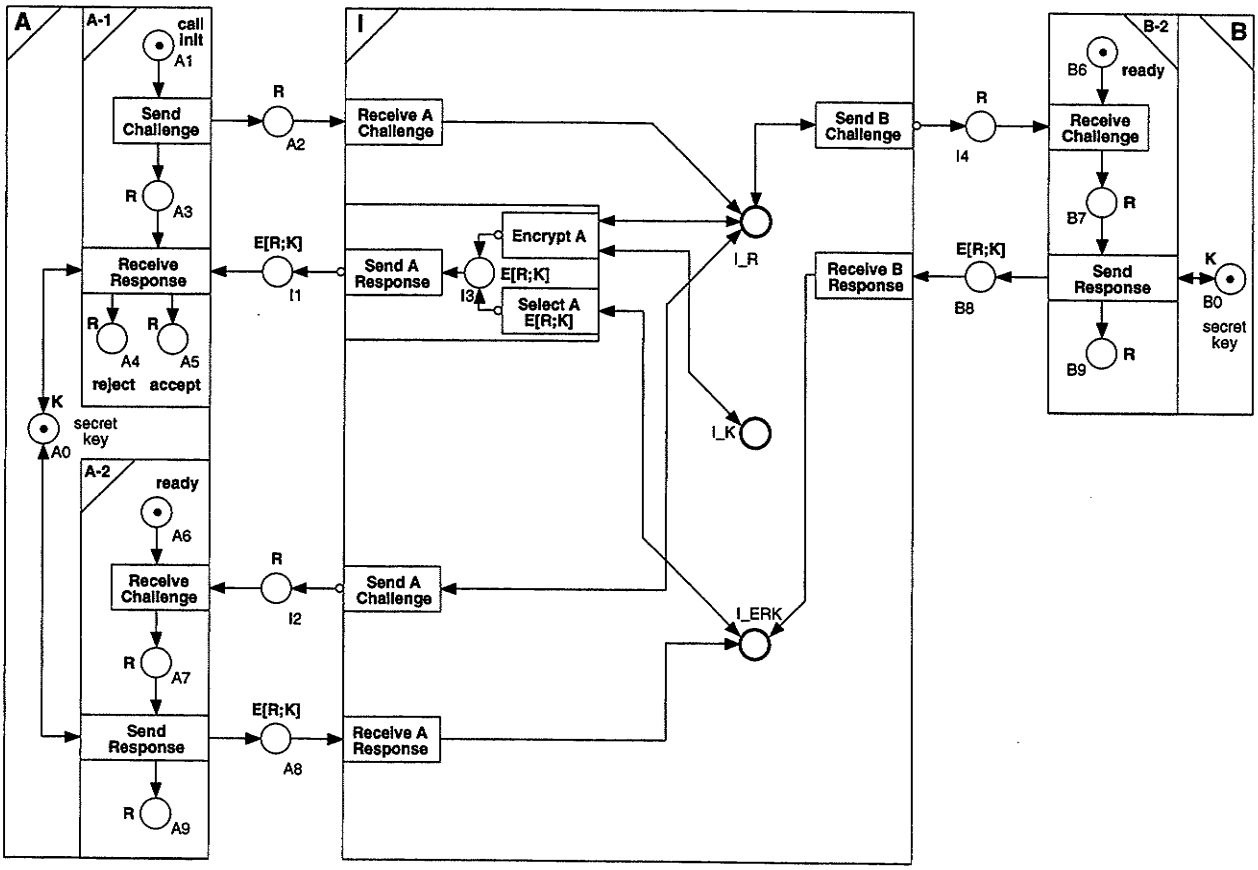

Figure 3. Petri net model of the one-way authentication protocol and intruder.

Random values for the generation of spurious messages are represented by the initial distribution of tokens with different values in the database places. For instance, I_R={r1,r2}, I_K={k1,k2}, and I_ERK={erk1,erk2} would give two random values of each type which the intruder would use to generate spurious messages.

2.3 State Analysis of the Petri Net Model

The state of the Petri net representation of the protocol is given by the distribution of tokens. In Figure 3, the initial state of the system is given as: A0={k}, A1={1}, A6={1}, B0={k}, and B6={1}. The value 1 denotes a single plain token. Tracing the states, through every possible permutation of transition firings results in a state-reachability tree. The reachability analysis involves examining the nodes of this tree, in particular the leaf nodes (i.e., the terminal states).

The process of traversing the reachability tree and gathering state information has been automated. Prolog is used as the language for experimentation, but the program can be mapped to a procedural language without too much effort. Each transition is specified by simple rule which describes the state transformations. The tree is traversed in breadth-first fashion which allows detection of duplicate nodes, and cuts the search effort dramatically. Each path is followed until a reject or accept state is reached, i.e., where there is a token in A4 or A5, respectively.

The reachability tree for the specification in Figure 3 is relatively shallow, 17 nodes; a reject or accept state is encountered after visiting 17 or fewer states. The number of unique states is 206, and the program completes execution in a few seconds. The quantities used in the analysis of the system are: the total number of terminal states, the number of accept states, and the number of accept states in which A5=B9. The condition for the latter quantity is a consequence of the objective of the one-way authentication protocol. A5=B9 indicates that the entity B did participate in the message exchange sometime during the execution of the protocol.

Since the set of accept states in which A5=B9 is a subset of the total number of accept states, it is necessary to show that the number of elements in each set is equal to determine that the objective of the protocol has not been violated. Given that the search of the state-space is exhaustive, the equality is also sufficient to claim that the protocol is secure under this analysis. This is based on the assumption that the model of the intruder accurately and completely specifies the capabilities of the true intruder.

In the case of Figure 3, the number of terminal nodes is 56, and the number accept states is 56. These numbers are equal, indicating that entity A never reaches a reject state. The number of accept states in which A5=B9 is 40. This is not equal to the total number of accept states; therefore, the paths leading to the 16 states in which A5≠B9 allow the intruder to subvert the objective of the

protocol. Examination of this set of states helps the designer determine the necessary protocol modifications.

Table 1a. Results from executing the protocol vulnerable to reflection attack.

Case (R,K,ERK)	Number of unique states	Number of terminal states	Number of accept states	Number of accept w/A5=B9	Time to execute
0 (0,0,0)	206	56	56 (7)	40 (5)	2 sec
1 (0,0,1)	646	242	84 (7)	60 (5)	8 sec
2 (0,1,0)	638	240	84 (7)	60 (5)	8 sec
3 (0,1,1)	1114	334	112 (7)	80 (5)	19 sec
4 (1,0,0)	2035	560	252 (13)	153 (8)	43 sec
5 (1,0,1)	5403	1641	369 (13)	225 (8)	242 sec
6 (1,1,0)	8634	2199	486 (13)	297 (8)	717 sec
7 (1,1,1)	13563	2821	603 (13)	369 (8)	1840 sec
8 (0,0,2)	1125	335	112 (7)	80 (5)	20 sec
9 (0,2,0)	1106	332	112 (7)	80 (5)	20 sec
10 (2,0,0)	9256	2370	672 (19)	384 (11)	607 sec
11 (3,0,0)	28259	6878	1400 (25)	775 (14)	5311 sec

Note: the numbers in parentheses indicate the number of equivalence classes.

Table 1b. Results from the modified protocol which prevents the reflection attack.

Case (R,K,ERK)	Number of unique states	Number of terminal states	Number of accept states	Number of accept w/A5=B9	Time to execute
0 (0,0,0)	242	72	36 (4)	36 (4)	3 sec
1 (0,0,1)	646	214	52 (4)	52 (4)	10 sec
2 (0,1,0)	638	212	52 (4)	52 (4)	10 sec
3 (0,1,1)	1110	294	68 (4)	68 (4)	24 sec
4 (1,0,0)	2187	582	144 (7)	144 (7)	57 sec
5 (1,0,1)	5445	1501	207 (7)	207 (7)	292 sec
6 (1,1,0)	8658	2005	270 (7)	270 (7)	863 sec
7 (1,1,1)	13585	2573	333 (7)	333 (7)	2230 sec
8 (0,0,2)	1121	295	68 (4)	68 (4)	24 sec
9 (0,2,0)	1102	292	68 (4)	68 (4)	24 sec
10 (2,0,0)	9694	2388	368 (10)	368 (10)	758 sec
11 (3,0,0)	29237	6876	750 (13)	750 (13)	6392 sec

Note: the numbers in parentheses indicate the number of equivalence classes.

The initial condition given in Figure 3 does not allow, however, for the existence of random values from which the intruder would generate spurious messages. A more complete analysis considers other cases with different initial conditions, adding one token at a time to each place in the database in the intruder model. The tests conducted thus far have included eleven different cases, and the results have been tabulated (see Table 1a). The most notable difference between cases is that the number of states, and correspondingly execution time, increases quite rapidly. Some measures have been taken to alleviate problems in this situation. By considering only a

subset of the places in the Petri net model, it is possible to group the terminal states into equivalence classes. In the tests, terminal states are considered to belong to the same class if they have the same values in corresponding places, excluding places I2, I3, and I4. The result is that there are fewer terminal states to consider, and that growth in the number of these classes of terminal states is stemmed. This provides hope that it may be proven that the intruder makes no more progress by increasing the number of spurious messages generated.

2.4 The Reflection Attack and Its Prevention

Table 2 lists all the classes of terminal states for a single case from the test set. The terminal states for which place B9 is empty are cause for concern. They indicate a successful reflection attack. The intruder is able to capture the challenge r from process A-1, redirect it to process A-2 to obtain a suitable encrypted value which it then feeds back to process A-1. This forces A-1 to accept but entity B never participated in the message exchange. The shaded rows in Table 2 are the suspect states which are eliminated by applying a simple modification to the protocol.

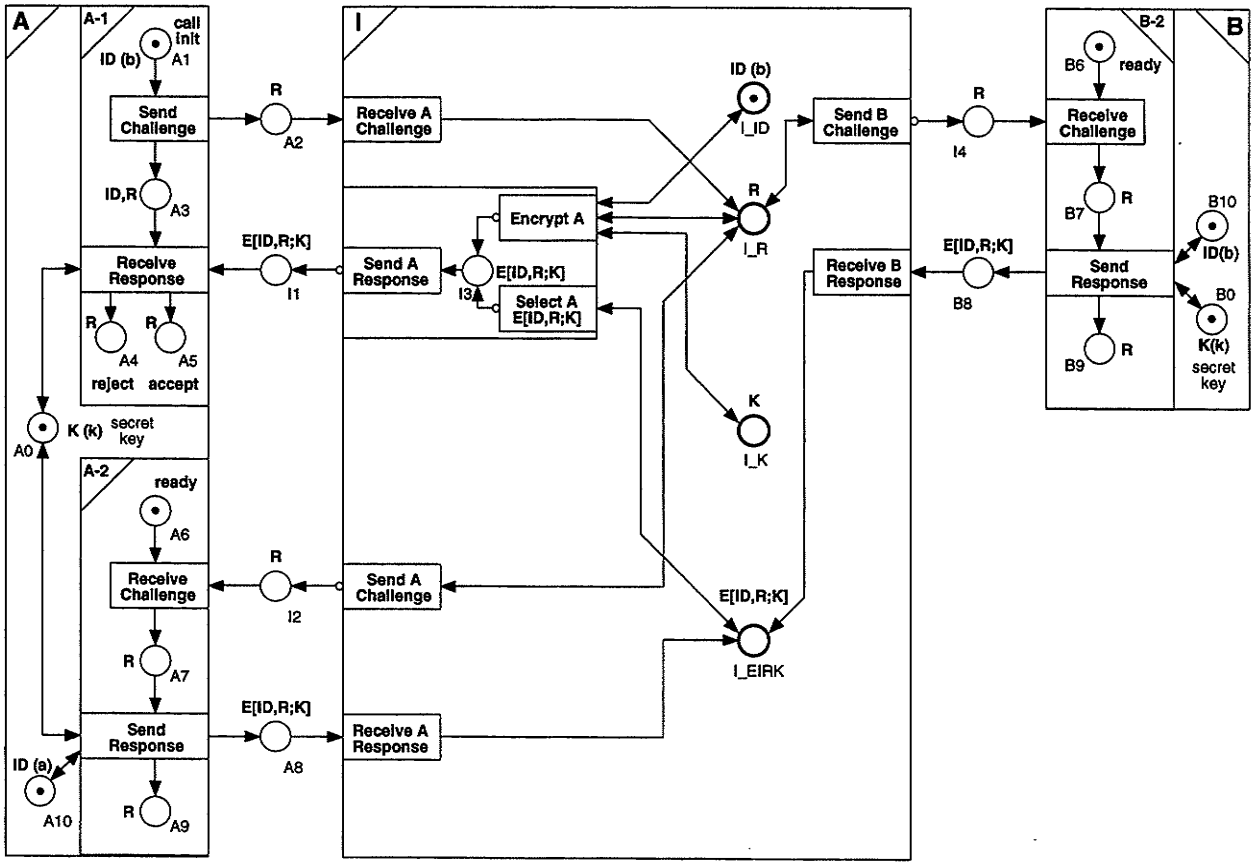

Figure 4. Petri net model of the modified one-way authentication protocol.

In the modified protocol given in Figure 4, the SEND RESPONSE function encrypts the entity identification (ID) along with the value of the challenge. This simple modification prevents the intruder from issuing a successful reflection attack. The results from Table 1a are updated to reflect the new protocol, and given in Table 1b.

Table 2. Results from the modified protocol which prevents the reflection attack.

	A6	A7	A8	A9	B6	B7	B8	B9	IR	IK	IERK
S1				r				r	r		e(r,k),e(r,k)
S2			e(r,k)	r				r	r		e(r,k)
S3				r			e(r,k)	r	r		e(r,k)
S4	r							r	r		e(r,k)
S5				r		r			r		e(r,k)
S6	1							r	r		e(r,k)
S7				r	1				r		e(r,k)

Note: for all states listed above A0=[k]; A1=A2=A3=A4=[]; A5=[r]; B0=[k].

3 Specifications for Other Protocols

3.1 Handset Authentication Protocol Used for CT2 and CT2Plus

The handset authentication protocol (a one-way authentication protocol) used in the CT2 and CT2Plus wireless communication protocols is specified by the Petri net in Figure 5; the network authentication protocol described in [18] is not considered here. Entities A and C represent distinct telepoint bases in different locations. Entity B is the handset attempting to authenticate itself to telepoint base #1 (entity A). The intruder is assumed to have same capabilities it had in the model of the simple one-way authentication protocol. In particular, in this wireless scenario, it may be able to intercept the signals (i.e., by jamming), and transmit its own. Note that the configuration used to analyze the protocol in section 2 is not applicable here, because the telepoint bases cannot execute the process used in the handset.

In Figure 5, labels in italics are given to relate the values back to the specification given in the CT2Plus documentation [17]. Some transitions also have additional labels which match the names of functions in the documentation. For example, the transition SEND CHALLENGE in the telepoint base matches the functions RNG (Random Number Generator), and F (the encryption function). The function f, which corresponds to the telepoint base transition RECEIVE ID, is described as proprietary to individual telepoint operators [16].

The goal of this protocol is to allow the handset to prove its identity to a given telepoint base. Each handset is assigned a unique identifier (id), and a PIN (the result of one-way function f operating on the id). The handset initiates the authentication protocol by transmitting its identifier to a designated telepoint base. The base passes the received id through function f to determine the

expected PIN of the handset, and responds by sending a challenge (a random value). The handset returns its PIN encrypted under the value of the challenge. If the telepoint base determines that the returned value matches the value of the expected PIN encrypted under the challenge by the base itself, the identity of the handset is accepted.

Two results follow from preliminary analysis of the protocol. First, it is found that the number of accept states in which A5≠B6 is zero. This demonstrates that telepoint base #1 (entity A) will not accept if entity B does not produce the expected response. The latter statement is a necessary condition to meet the goal of the protocol. Second, the discovery that the number of terminal states in which C5 has a token is 80 (i.e., non-zero), uncovers a well known attack on all

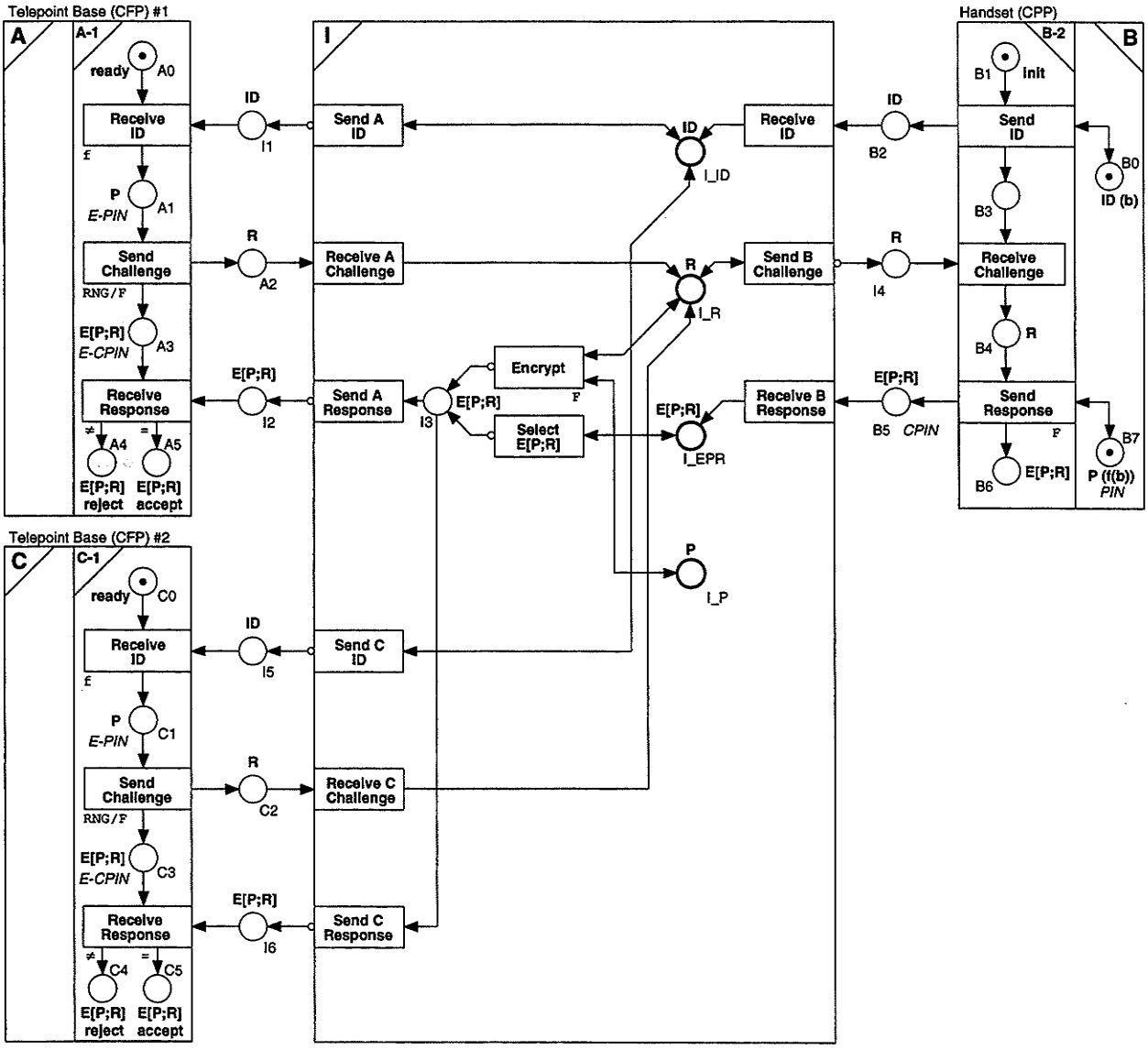

Figure 5. Petri net model of the handset authentication protocol used in CT2 and CT2Plus.

protocols in which the intruder has control of the communication channels. In this attack, the intruder simply redirects messages from entity B to entity C rather than A, and the replies from C back to B. It forces the wrong telepoint base to authenticate B. Without additional precautions, the intruder may be able to continue accessing services under the identity of entity B through telepoint base #2 (entity C). Subsequent steps in CT2 and CT2Plus, however, can make this attack infeasible. Table 3 summaries the results.

Table 3. Results from the modified protocol which prevents the reflection attack.

Case (R,K,ERK)	# unique states	# terminal states	# accept states	# accept w/A5≠B6	# terminal w/C5≠∅
0 (0,0,0)	1650	384	224	0	80

4 Conclusions

Coloured Petri nets have proven to be effective for both the specification and our iterative analysis and design approach for cryptographic protocols. Using a common model throughout contributes to reliable, accurate results. Rules for the construction of an intruder model have been given. By inserting this intruder into the protocol specification, forward state analysis exhaustively checks all possible intruder actions. Software developed to automate the analysis has made it feasible to analyze a variety of protocols.

With Petri Net Objects, multiple-iteration and parallel session attacks [13] can now be modeled. A previously known reflection attack on the simple one-way authentication protocol was given by our analysis. Further analysis showed that a slight protocol modification eliminated the threat of this attack. Another Petri net model captured a common attack on two-party protocols.

5 Future Work

Further study of the implications of the terminal states with respect to the security properties and objectives of a cryptographic protocol is necessary. This will allow further generalization of our approach, and facilitate the development of a more flexible analysis tool. It is anticipated that we will move from Prolog to an object-oriented language such as C++ for the tool since the PNO model lends itself to an object-oriented implementation.

More protocols will be analyzed with particular attention to wireless communication systems. Cellular and cordless applications are particularly vulnerable to intruder attacks, and represent an area of considerable concern. The objectives of the protocols to be examined will encompass authentication and privacy.

6 Acknowledgment

Thanks are extended to Paul van Oorschot for his helpful suggestions.

Bibliography

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

[2] C. Boyd and W. Mao. "On a limitation of BAN logic". *Advances in Cryptology—Proceedings of EUROCRYPT '93*, Lecture Notes in Computer Science 765, Springer-Verlag, Berlin, 1993, pp. 240-247.

[3] R. Kemmerer, C. Meadows and J. Millen. "Three Systems for Cryptographic Protocol Analysis". *Journal of Cryptology*, 1994, 7:79-130.

[4] M. Diaz, (tutorial). "Modeling and Analysis of Communication and Cooperation Protocols Using Petri Net Based Models". *2nd International Workshop on Protocol Specification, Testing and Verification*, Idyllwid, CA, May 1982.

[5] M. Diaz and P. Azema. "Petri net based models for the specification and validation of protocols". *Advances in Petri Nets 1984*, Lecture Notes in Computer Science 188, Springer-Verlag, Berlin,, 1985 pp. 101-121.

[6] Tadao Murata. "Petri Nets: Properties, Analysis and Applications". *Proceedings of the IEEE*, April 1989, 77(4):541-580.

[7] Kurt Jensen. "Coloured Petri Nets: A High Level Language for System Design and Analysis". *Advances in Petri Nets 1990*, Lecture Notes in Computer Science 483, Springer-Verlag, Berlin, 1991, pp. 342-416.

[8] N. Behki. *An integrated approach to protocol design.* Master's Thesis, Electrical Engineering, Queen's University, Kingston, Ontario, 1990.

[9] N. Behki, and S.E. Tavares. "An integrated approach to protocol design". *Proceedings of the 1989 IEEE Pacific Rim Conference on Computers, Communications and Signal Processing*, May 30-June 2, 1989, pp. 244-248.

[10] Darryl M. Stal. *Backward State Analysis of Cryptographic Protocols Using Coloured Petri Nets.* Master's Thesis, Electrical and Computer Engineering, Queen's University, Kingston, Ontario, 1994.

[11] C.M. Morton. *A Modular Approach to Modeling Cryptographic Protocols Using Petri Nets.* Master's Thesis, Electrical Engineering, Queen's University, Kingston, Ontario, 1993.

[12] L.C. Robart. *Decomposition Techniques for Cryptographic Protocols in Wireless Communication Systems.* Master's Thesis, Electrical Engineering, Queen's University, Kingston, Ontario, 1993.

[13] R. Bird, I. Gopal, A. Herzberg, et al. "Systematic Design of Two-Party Authentication Protocols", *Advances in Cryptology—CRYPTO '91*, Lecture Notes in Computer Science 576 Springer-Verlag, Berlin, 1992, pp. 43-59.

[14] C. Mitchell. "Limitations of Challenge-Response Entity Authentication". Electronic Letters, August 17, 1989, Volume 25, Number 17, pp. 1195-96.

[15] Whitfield Diffie, Paul van Oorschot, and Michael J. Wiener. "Authentication and Authenticated Key Exchanges". *Designs, Codes and Cryptography*, Kluwer Academic Publishers, The Netherlands, 1992, 2:107-125.

[16] RES Technical Commitee, ETSI. *CT2 Common Air Interface: Common air interface specification to be used for the internetworking between cordless telephone apparatus in the frequency band 864.1 MHz to 868.1 MHz, including public access services*, version 1.1, June 30 1991.

[17] Submission to the Radio Advisory Board of Canada Industry Advisory Committee Working Group on Radio Interface Standards. *CT2Plus: A Proposal for a Canadian Common Air Interface Standard*, Issue 2.0, December 1990.

[18] RES Technical Commitee, ETSI. *Common air interface specification to be used for the internetworking between cordless telephone apparatus in the frequency band 864.1 MHz to 868.1 MHz, including public access services*, draft, 2nd edition, January 31, 1994.

An Efficient and Secure Authentication Protocol Using Uncertified Keys*

I-Lung Kao and Randy Chow
Department of Computer and Information Sciences
University of Florida
Gainesville, Florida 32611
{kao,chow}@cis.ufl.edu

Abstract

Most authentication protocols in distributed systems achieve identification and key distributions on the belief that the use of a uncertified key, i.e. the key whose freshness and authenticity cannot be immediately verified by its receiving principal while being received, should be avoided during the midway of an authentication process. In this paper we claim that using a uncertified key prudently can give performance advantages and not necessarily reduces the security of authentication protocols, as long as the validity of the key can be verified at the end of an authentication process. A nonce-based authentication protocol using uncertified keys is proposed. Its total number of messages is shown to be the minimal of all authentication protocols with the same formalized goals of authentication. The properties which make the protocol optimal in terms of message complexity are elaborated, and a formal logical analysis to the protocol is performed. The protocol is extended to prevent the session key compromise problem and to support repeated authentication, in a more secure and flexible way without losing its optimality.

1 Introduction

Authentication in essence is a process of verifying the authenticity of one's claim about its identity. It is one of the most important aspects of computer security, since other security services such as authorization, accounting, and auditing are all based upon it. In a distributed computing environment with machines connected by vulnerable network links, any two principals on different machines need to authenticate each other first on communication initiation such that an intruder cannot impersonate a principal to the other. Furthermore, distributed applications frequently require that the messages transmitted over the network be confidential only to the communicating peers (e.g. on-line credit card payment). Since encryption is currently the main technique to achieve this requirement, at least a *session key* needs to be distributed first between two communicating principals before a session of confidential data transmission between them can initiate. This session key is also used to provide *message origin authentication* during data communication following an authentication process. That is, any message encrypted with the session key after authentication is believed to originate from the peer principal who holds the session key. Thus, the distribution of a session key is often carried out concurrently with the authentication process.

In general, authentication protocols for distributed systems can be divided into two categories depending upon how the freshness of key-distribution messages is determined. One category of protocols uses nonces and challenge/response exchanges to verify if the response to a key distribution request is fresh or not. Since replay attacks can be effectively prevented by the use of nonces, most proposed authentication protocols are nonce-based [8, 11, 12, 13, 14, 17]. The other category of protocols uses timestamps to ensure the freshness of messages and must assume that all machines are properly clock-synchronized [9]. The number of messages required by timestamp-based protocols can be reduced since no round-trip traffic is required to guarantee message freshness as in the case of nonce-based protocols. Kerberos [15] and KryptoKnight [10] are the most well-know authentication services based on timestamps and nonces, respectively. Due to the possible imperfection of clock synchronization mechanisms, timestamp-based pro-

*This research is partially supported by U.S. Army under contract DASG60-94-C-0076.

tocols are vulnerable to both conventional copy-and-replay and suppress-and-play attacks discussed by Gong [6]. Therefore, we will only address nonce-based protocols in the following.

After the initial authentication is established and a communication session has been completed by a pair of principals, there may be needs for future communication sessions. If it is believed that the session key has not been compromised, authentication for subsequent sessions can be accomplished by using the idea of repeated authentication, without going through the authentication server. Effectively, the load of the authentication server for key generation and distribution can be reduced. The KSL protocol proposed by Kehne, Schonwalder, and Langendorfer [8] is an example of nonce-based protocols for both initial and repeated authentications. Its initial authentication requires only five messages and the subsequent repeated authentications require three messages for each session. Later, Neuman and Stubblebine [13] presented another nonce-based protocol which requires only four messages for the initial authentication and still three messages for each subsequent authentication. However, it offers better protocol efficiency by sacrificing the security of the protocol, in that a weaker set of formalized goals [3] is achieved than that achieved by the KSL protocol. More specifically, the Neuman-Stubblebine protocol lacks a final belief reached by the KSL protocol: principal A is convinced that his communicating peer, principal B, also trusts the session key to be used between them [13, 16]. The primary objective of the paper is to propose an new authentication protocol that achieves both the lower message overhead of the Neuman-Stubblebine protocol and the stronger authentication goals of the KSL protocol. The proposed protocol is further extended to prevent the compromise of session keys, an issue not addressed by either protocol. Support of repeated authentication is then incorporated to the protocol without losing its performance advantages.

2 The Proposed Nonce-based Protocol

The assumptions of the environment where the protocol is to be operated and of possible attacks are basically the same as those assumed by most existing authentication protocols. Two principals, A and B, wishing to authenticate each other and to obtain a shared session key for subsequent communication. A trusted authentication server S shares a *master key* with each principal and is capable of producing good session keys and sending them securely on the requests of principals. No clock synchronization among machines is assumed, so nonce-based challenge/response exchanges are used to guarantee the freshness of messages.

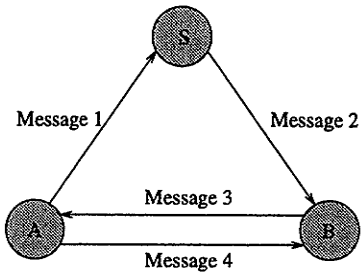

Figure 1: A nonce-based authentication protocol

The message flow of the protocol is shown in Figure 1 and the contents of each message is as follows:

Message 1	$A \rightarrow S:$	A, B, N_a
Message 2	$S \rightarrow B:$	$\{A, B, N_a, K_{ab}\}_{K_{as}}$, $\{A, B, N_a, K_{ab}\}_{K_{bs}}$
Message 3	$B \rightarrow A:$	$\{A, B, N_a, K_{ab}\}_{K_{as}}$, $\{N_a\}_{K_{ab}}, N_b$
Message 4	$A \rightarrow B:$	$\{N_b\}_{K_{ab}}$

Principal A initiates the authentication by sending S a plaintext message containing the identities of itself and the desired communicating peer B, and a nonce N_a (message 1). After S receives this message, he generates a session key K_{ab} and appends it to the identities of both parties and nonce N_a to form two credentials, one for A and the other for B. Both credentials have exactly the same contents, but one is encrypted with A's master key K_{as}, and the other is encrypted with B's master key K_{bs}. S sends both credentials to B (message 2), who then decrypts the second one and finds out that A wants to authenticate with B mutually, N_a is the nonce issued by A, and K_{ab} is generated by S to be used as a session key for future communication between A and B. B then forwards the first credential from S to A, and also sends an encrypted N_a with K_{ab} and another nonce N_b (message 3). Upon receiving them, A decrypts the credential to get K_{ab} and verifies its freshness by checking the presence of N_a. A also authenticates B by decrypting the encrypted part with K_{ab} and comparing the result with N_a. If they match, A encrypts N_b with K_{ab} and sends it back to B (massage 4) to prove its identity to B.

In the protocol, A verifies the identity of B by checking whether the peer principal is able to encrypt nonce N_a with session key K_{ab}. This verifi-

cation is based upon two beliefs of A. The first one is that on the request of authentication (message 1), S will issue a credential containing N_a and K_{ab} and encrypted with K_{bs} for principal B (message 2). The second belief of A is that only S and B share master key K_{bs}, so no other principal except B is able to send the encrypted N_a with K_{ab} (message 3). Therefore, the protocol prevents against impersonation of B by the assumptions of the correct behavior of the authentication server and of the secrecy of master keys. Furthermore, since nonce N_a is used only for the current session, replay of old messages issued by either S or B will be detected.

On the other side, B verifies the identity of A by the use of uncertified session keys. When B receives message 2, he has no way to tell whether the message is either a replay or an impersonation attempt initiated by a malicious principal C. B only presumes that some principal who claims to be A wants to authenticate each other with himself *for the current session*. To verify message 2 is authentic and fresh, B needs to use (and temporarily believes) the uncertified session key K_{ab} to encrypt N_a and also sends its own nonce N_b in the clear. If A returns message 4 as expected, B believes in the authenticity and freshness of K_{ab}. If message 2 is only a replay (either copy-and-replay or suppress-and-play), A will detect it and thus will not respond with a normal message 4 (instead, A probably sends back an error message to inform B that a replay is possibly occurring), so B knows K_{ab} is not fresh. If principal C wants to impersonate A and initiates an authentication process, he is incapable of producing message 4 since K_{ab} is confidential to C. Therefore, B can verify its temporal belief on the authenticity and freshness of message 2 by a nonce challenge/response exchange with A. Note that B authenticates A basing upon the beliefs similar to those on which A bases to authenticate B.

3 A Formal Protocol Analysis Using BAN Logic

To describe the protocol formally, each message of the protocol is presented in an idealized form:

$$\begin{array}{lll} \text{Message 2} & S \rightarrow B: & \{N_a, A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{as}}, \\ & & \{N_a, A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{bs}} \\ \text{Message 3} & B \rightarrow A: & \{N_a, A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{as}}, \\ & & \{N_a, A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{ab}} \\ \text{Message 4} & A \rightarrow B: & \{N_b, A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{ab}} \end{array}$$

The first message is omitted since it is in the clear and thus provides no guarantee about the properties of the protocol. The result is as if S acted spontaneously. Message 2 expresses that both credentials from S contain nonce N_a and session key K_{ab} to be shared between A and B. The first component of message 3 indicates that B faithfully forwards the first component of message 2 to A, and the second component means that B temporarily trusts K_{ab}, and uses it to encrypt N_a to imply to A that it would like to share K_{ab} with A upon subsequent verification. The last message indicates that A has verified the freshness of K_{ab}, and responds to B's challenge by encrypting N_b with K_{ab}.

The initial assumptions of the protocol in BAN Logic notation are:

$$\begin{array}{ll} \textit{Key} & 1.A \models A \overset{K_{as}}{\leftrightarrow} S \\ & 2.B \models B \overset{K_{bs}}{\leftrightarrow} S \\ & 3.S \models A \overset{K_{as}}{\leftrightarrow} S \\ & 4.S \models B \overset{K_{bs}}{\leftrightarrow} S \\ & 5.S \models A \overset{K_{ab}}{\leftrightarrow} B \\ \\ \textit{Server} & 1.A \models (S \Rightarrow A \overset{K}{\leftrightarrow} B) \\ & 2.B \models (S \Rightarrow A \overset{K}{\leftrightarrow} B) \\ \\ \textit{Freshness} & 1.A \models \sharp(N_a) \\ & 2.B \models \sharp(N_b) \\ & 3.B \models \sharp(A \overset{K}{\leftrightarrow} B) \end{array}$$

The first four assumptions in the *Key* group specify the initial beliefs on the secrecy of master keys between the principals and the authentication server. The fifth denotes that session key K_{ab} can only be generated by S. The next group (*Server*) indicates the trusts that A and B have on the server to generate a good session key. The last group of assumptions is about the freshness of nonces and keys. The first two indicate that each principal can issue a nonce and trusts only the nonce issued by himself. The last one is needed by B for attempting to use a uncertified key. As pointed out in the BAN Logic paper about the Needham-Schroeder protocol [3], the last assumption is not as obvious as others initially, but can be verified later by the protocol itself.

The formal proof of the protocol using the postulates of BAN Logic is presented as follow. First, A sends S a cleartext message containing a nonce. S then sends message 2 to B, that is:

$$B \triangleleft \{N_a, A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{as}}, \{N_a, A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{bs}}$$

B can decrypt the second component of this mes-

sage with K_{bs}. Applying the message-meaning rule to it, we can deduce:

$$B \models S \mid\sim (A \overset{K_{ab}}{\leftrightarrow} B)$$

With the application of the nonce-verification rule to the above assertion and the assumption $B \models \sharp(A \overset{K}{\leftrightarrow} B)$, we obtain:

$$B \models S \models A \overset{K_{ab}}{\leftrightarrow} B$$

With the jurisdiction rule, we immediately get:

$$B \models A \overset{K_{ab}}{\leftrightarrow} B$$

B, temporarily trusting K_{ab}, generates message 3 and sends it to A, thus:

$$A \triangleleft \{N_a, A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{as}}, \{N_a, A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{ab}}$$

A can decrypt the first component encrypted with K_{as}. Since S knows N_a to be fresh, we can apply the message-meaning rule, leading to:

$$A \models S \mid\sim (A \overset{K_{ab}}{\leftrightarrow} B)$$

Applying the nonce-verification and jurisdiction rules in a way similar to the above described, we obtain:

$$A \models A \overset{K_{ab}}{\leftrightarrow} B$$

After getting K_{ab}, A uses it to decrypt the second component of message 3 and checks the presence of N_a. Therefore, the message-meaning rule applies:

$$A \models B \mid\sim (A \overset{K_{ab}}{\leftrightarrow} B)$$

With the nonce-jurisdiction rule, we can obtain:

$$A \models B \models A \overset{K_{ab}}{\leftrightarrow} B$$

Then A replies B with message 4. B deduces from the message that A believes in the session key. With an analysis similar to the one applied to the second component of message 3, we can get:

$$B \models A \models A \overset{K_{ab}}{\leftrightarrow} B$$

In conclusion, the final beliefs of both principals achieved by this protocol are:

$$A \models A \overset{K_{ab}}{\leftrightarrow} B \qquad B \models A \overset{K_{ab}}{\leftrightarrow} B$$
$$A \models B \models A \overset{K_{ab}}{\leftrightarrow} B \qquad B \models A \models A \overset{K_{ab}}{\leftrightarrow} B$$

which are exactly the formalized goals of authentication for all authentication protocols as recommended by the authors of BAN Logic. It should be noticed that these goals can be achieved with only four messages.

4 Countering Session Key Compromises

Like the original Needham-Schroeder protocol [11], the final beliefs of our protocol are reached assuming that B accepts the session key as new upon receiving it, though the assumption can be verified as the protocol proceeds. Not surprisingly, our protocol is also vulnerable to the session key compromise attack as pointed out by Denning and Sacco [4] to the original Needham-Schroeder protocol. That is, if an intruder C compromised an old session key and copied messages 2 and 4 of the protocol run in which the session key was used, he can pretend to B as if he were A . B is incapable, by the protocol itself, of knowing whether a session key has been compromised or not. Note that message 4 of the protocol only verifies to B whether the session key is a replay or the result of an impersonation attempt, *if the key is not compromised.*

This possible attack can be prevented by including timestamps in messages as suggested by Denning and Sacco [4], which requires clock synchronization of all the machines. Alternatively, the solution proposed by Needham and Schroeder [12] for their original protocol requires B to generate its own nonce initially and S to include this nonce in the message containing the session key. This unfortunately leads to at least two more messages in a protocol run. The following describes an enhancement of our proposed protocol to counter this impersonation attack, requiring neither time synchronization nor additional messages.

Without taking consideration of the robustness of cryptographic algorithms and the possibility of brute-force cryptanalysis, session keys are more easier to be compromised than master keys because of operation reasons. Session keys are used over a relatively longer time period and are usually stored in (probably insecure) local memory or registers for efficient encryption and decryption for the entire communication session. In general, the attacks to session keys can be effectively prevented by raising the quality of session keys (e.g. using longer keys) or improving the protocol itself to reduce the vulnerability resulting from insecure local memory and communication links. The strategy we take is to have S issue another key K_t, along with K_{ab}, to the principals in the protocol. K_t is used just for the current authentication session and is discarded immediately after authentication. The improved protocol becomes:

Message 1	$A \rightarrow S:$	A, B, N_a
Message 2	$S \rightarrow B:$	$\{A, B, N_a, K_{ab}, K_t\}_{K_{as}}$, $\{A, B, N_a, K_{ab}, K_t\}_{K_{bs}}$
Message 3	$B \rightarrow A:$	$\{A, B, N_a, K_{ab}, K_t\}_{K_{as}}$, $\{N_a, K_{ab}\}_{K_t}, N_b$
Message 4	$A \rightarrow B:$	$\{N_b, K_{ab}\}_{K_t}$

K_t is issued by S and included in both credentials of message 2. It is used by B to encrypt N_a and K_{ab} to tell A that B temporarily trusts both keys

K_{ab} and K_t. A verifies B's temporal trusts on these keys by checking the presence of N_a within the first credential in message 3, and sends back to B the encrypted N_b and K_{ab} with key K_t. After K_t is used by A for encrypting message 4, it is removed right away from the local memory of A's machine. After message 4 is received and verified, B also immediately removes K_t from its local memory.

The use of K_t is exclusively for authentication only. A new K_t is generated by the authentication server S for each initial mutual authentication. An intruder may have compromised K_{ab} and replays old authentication messages, but will fail to impersonate A in a run of this improved protocol, since he is unable to encrypt message 4 with the new K_t. K_t is much more difficult to break than K_{ab} because it is used by (and meaningful to) A and B only once, for a very brief period. Another advantage of this improved protocol is that the same K_{ab} can be used for multiple sessions, since each session initiation is checked by a different K_t.

5 Repeated Authentication

Authentication servers heavily utilized may become a performance and security bottleneck in the system. If a system operates in a relatively benign environment and the session keys distributed possess pretty good quality, it is possible to reduce the workload of authentication servers and the corresponding communication overhead by repeating the use of a previous session key for subsequent authentication sessions. Protocols for repeated authentication usually distribute some credentials (which are often called *tickets* and will be referred to as *session-key certificates* in this paper) to principals during an initial authentication session. In a subsequent authentication session, a session-key certificate is used to securely convey a session key distributed earlier to the principal who can recognize that certificate, without the need to contact the authentication server again. In the following we show how our protocol can be extended to deal with repeated authentication, in a more secure and symmetrical way than the KSL and Neuman-Stubblebine protocols.

Initial authentication: getting session-key certificates

Messages 3 and 4 in the initial authentication protocol are further extended to include session-key certificates for repeated authentication.

Message 1	$A \rightarrow S:$	A, B, N_a
Message 2	$S \rightarrow B:$	$\{A, B, N_a, K_{ab}, K_t\}_{K_{as}}$, $\{A, B, N_a, K_{ab}, K_t\}_{K_{bs}}$
Message 3	$B \rightarrow A:$	$\{A, B, N_a, K_{ab}, K_t\}_{K_{as}}$, $\{N_a, K_{ab}\}_{K_t}, N_b$, $\{A, B, T_b, K_{ab}\}_{K_{bs}}$
Message 4	$A \rightarrow B:$	$\{N_b, K_{ab}\}_{K_t}$, $\{A, B, T_a, K_{ab}\}_{K_{as}}$

In message 3, B also sends A a session-key certificate which contains the identities of both A and B, session key K_{ab}, and a generalized timestamp T_b, suggested by the KSL protocol, and is encrypted with the master key of B. After checking the validity of message 3, A also returns B with a session-key certificate which contains similar information but is encrypted with A's own master key. Since a session-key certificate is encrypted with the master key of its issuer, it is only recognizable to the issuer. The purpose of a generalized timestamp is to limit the validity of a certificate, corresponding to the *local time* of the issuer. Therefore, the assumption of global clock synchronization is not required by using timestamps this way.

Subsequent authentication: exchanging the certificates

After the initial authentication, A and B hold session-key certificates for each other. When the communication session between A and B following the initial authentication is completed, K_{ab} is removed from the local memory of both principals' machines. Since the session key does not need to be kept in the memory of principal A's machine after an initial communication session as in the KSL and Neuman-Stubblebine protocols, this method of protecting session keys is more secure than those protocols. It also distributes the risk of compromising all the session keys of A at the same time if A is communicating with multiple peer principals, since each session-key certificate of A is held by a distinct principal. When A wants to repeat an authentication with B next time, he initiates a protocol as follows:

Message 1'	$A \rightarrow B:$	$\{A, B, T_b, K_{ab}\}_{K_{bs}}, N'_a$
Message 2'	$B \rightarrow A:$	$\{A, B, T_a, K_{ab}\}_{K_{as}}$, $\{N'_a\}_{K_{ab}}, N'_b$
Message 3'	$A \rightarrow B:$	$\{N'_b, N'_a\}_{K_{ab}}$

A sends the session-key certificate previously issued by B and nonce N'_a in message 1'. After verifying that the certificate is still fresh, B temporarily trusts session key K_{ab} and uses it to encrypt N'_a. B then sends back the matching session-key certificate

issued earlier by A, the encrypted N'_a, and a new nonce N'_b (message 2'). The last message shows that A has already trusted K_{ab} and verified the identity of B. Upon receiving it, B verifies its trust on K_{ab} and the identity of A.

The subsequent authentication protocol is actually similar in both spirit and style to the initial authentication protocol. The difference between them is that in the former two principals exchanges session-key certificates originally generated by each other, and in the latter A initiates S to generate a session key for both principals and requires B to forward the session key to himself. It should be noticed that possession of a session-key certificate only means holding some key information for another principal. It does not provide any authentication guarantee. The capability of recognizing (decrypting) the certificate and then encrypting a nonce with the session key is still needed to verify the identity of a principal. Note also that even in the initial authentication protocol B sends a session-key certificate to A (message 3) prior to verifying the session key, he will not accept the session-key certificate from A as valid if the nonce response from A is different from the one expected.

In addition to protecting the session keys more securely, another advantage of our repeated authentication protocol over the KSL and Neuman-Stubblebine protocols is that either A or B can initiate a subsequent authentication. The subsequent authentication protocol initiated by B is symmetrical to the one shown above. Both the KSL and Neuman-Stubblebine protocols presume the role of A as a client and the role of B as a server in an initial authentication, and their roles will not change during subsequent authentications. However, our protocol does not assume the role of any principal, rendering more flexibility on who can initiate a subsequent authentication. In modern client-server type distributed systems, principal B, being a server of client A in the current communication session, could be a client of A as a server in the next session. Due to these reasons, our authentication protocol is more adaptive to another distributed system paradigm, the peer-to-peer communication style.

Preventing oracle session attacks

Encrypting N'_a in the last message of a repeated authentication provides an association between both message 2' and 3' in the same protocol run. Its main purpose is to prevent the oracle session attack [2], in which an intruder starts two separate authentication sessions with principals A and

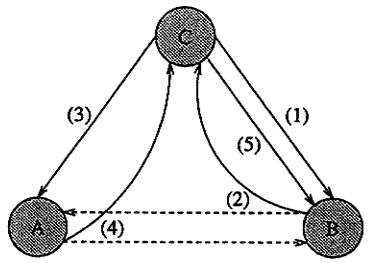

Figure 2: An oracle session attack by an intruder C

B such that he can utilize the messages in one session to impersonate a principal successfully in the other session.

Let's demonstrate an attack scenario with a version of our repeated authentication protocol without encrypting N'_a in message 3' (Figure 2):

(1) $C \rightarrow B$: $\{A, B, T_b, K_{ab}\}_{K_{bs}}, N'_c$
(2) $B \rightarrow A$: $\{A, B, T_a, K_{ab}\}_{K_{as}}, \{N'_c\}_{K_{ab}}, N'_b$
——— intercepted by C
(3) $C \rightarrow A$: $\{A, B, T_b, K_{ab}\}_{K_{as}}, N'_b$
(4) $A \rightarrow B$: $\{A, B, T_b, K_{ab}\}_{K_{bs}}, \{N'_b\}_{K_{ab}}, N'_a$
——— intercepted by C
(5) $C \rightarrow B$: $\{N'_b\}_{K_{ab}}$

An intruder C, who has copied a session-key certificate $\{A, B, T_b, K_{ab}\}_{K_{bs}}$ during an initial authentication or an earlier repeated authentication session, pretends to be A by sending B that certificate and nonce N'_c. B thinks that this authentication request was from A and responds with A's session-key certificate $\{A, B, T_a, K_{ab}\}_{K_{as}}$, nonce response $\{N'_a\}_{K_{ab}}$, and a new nonce N'_b. All are intercepted by C. Then C pretends to be B by sending A (the oracle) the certificate and nonce N'_b that he just obtained from B. A also thinks that this authentication request was from B and responds with B's session-key certificate $\{A, B, T_b, K_{ab}\}_{K_{bs}}$, nonce response $\{N'_b\}_{K_{ab}}$, and a new nonce N'_a. This message again is intercepted by C. C thus can impersonate A successfully by just passing the nonce response $\{N'_b\}_{K_{ab}}$ to B. Although K_{ab} is not compromised, A's privileges still could be abused by C by just replaying some encrypted messages (also encrypted with K_{ab} intercepted in an earlier communication session).

This type of attacks can succeed if there is no explicit association between messages 2' and 3'. With N'_a encrypted in both messages, it is ensured that message 3' obtained by B belongs to the same protocol run as message 2' he has sent. This technique is a realization of the suggestion by some authentication protocol researchers [1, 5] that *messages in a*

particular protocol run should be logically linked in a manner such that the re-use of messages from a previous run or the introduction of messages from a concurrent run can be detected.

Logical analysis and timestamp

Different from the initial authentication protocol, a principal running the protocol for subsequent authentication checks the freshness of a session key by using the generalized timestamp associated with it. However, even with a generalized timestamp, a principal still cannot tell whether or not the sending of a session-key certificate by the peer principal is a replay or an impersonation attempt, if the lifetime of the certificate has not ended yet. Using generalized timestamps this way actually does not guarantee message freshness as effective as provided by nonce challenges. A principal still needs to verify that an authentication message is fresh by a nonce handshaking with his communicating peer.

The generalized timestamp that represents the lifetime of a session-key is solely determined by the issuer of the certificate. This autonomy may result in a pair of related certificates with very different lifetimes. This timestamp discrepancy problem can be easily solved by including some timestamp information in the second component (encrypted with K_t) of message 3 in the initial authentication protocol. When the receiver principal obtains this information, he can refer to it for determining the timestamp parameters of the session-key certificate to be issued by himself in message 4. Because no full negotiation between both principals about the timestamps is performed (actually not a necessity) and it is meaningless for a principal to issue a certificate which lives longer than that issued by his peer principal, it tends to make the certificate issued by the principal sending message 4 expire earlier.

The extended protocol has also been analyzed by using BAN logic, and the four formalized goals of authentication can also be achieved [7].

6 Conclusion

The paper presents a new nonce-based authentication protocol which makes use of uncertified keys to reduce its message complexity. The protocol is formally shown to achieve the authentication goals as recommended by BAN. The messages required for the initial authentication is four and three for each subsequent repeated authentication. The protocol is improved to become more robust against impersonation attacks in later authentications even when session keys are compromised. It is achieved by using an additional one-time key without increasing the number of messages during initial authentication. The protocol is further extended for repeated authentication. The use of symmetrical storing of session-key certificates is more secure and adaptive to the peer-to-peer communication paradigm in distributed systems. Some implementation issues are under studies.

References

[1] Martin Abadi and Roger Needham, *Prudent Engineering Practice for Cryptographic Protocols*, DEC SRC Research Report 125, June 1994.

[2] Ray Bird, et al., "Systematic Design of a Family of Attack-Resistant Authentication Protocols," *IEEE Journal on Selected Areas in Communications*, Vol. 11, No. 5, June 1993, pp. 679 – 693.

[3] Michael Burrows, et al., *A Logic of Authentication*, DEC SRC Research Report 39, February 1990.

[4] Dorothy E. Denning and Giovanni Maria Sacco, "Timestamps in Key Distribution Protocols," *Communications of the ACM*, Vol. 24, No. 8, August 1981, pp. 533 – 536.

[5] Whitfield Diffie, et al., "Authentication and Authenticated Key Exchanges," *Design, Codes and Cryptography*, Vol. 2, No. 2, June 1992, pp. 107 – 125.

[6] Li Gong, "A Security Risk of Depending on Synchronized Clocks," *Operating Systems Review*, Vol. 26, No. 1, January 1992, pp. 49 – 53.

[7] I-Lung Kao and Randy Chow, "An Efficient and Secure Authentication Protocol Using Uncertified Keys," Technical Report UF-CIS-TR95-008, University of Florida, February, 1995.

[8] A. Kehne, et al., "A Nonce-Based Protocol for Multiple Authentication," *Operating Systems Review*, Vol. 26, No. 4, October 1992, pp. 84 – 89.

[9] Barbara Liskov, "Practical Uses of Synchronized Clocks in Distributed Systems," *Proceedings of the tenth Annual ACM Symposium on Principles of Distributed Computing*, Montreal, Quebec, Canada, August 1991, pp. 1 – 9.

[10] Refik Molva, et al., "KryptoKnight Authentication and Key Distribution System," *Proceedings of 1992 European Symposium on Research in Computer Security*, Toulouse, France, November 1992.

[11] Roger M. Needham and Michael D. Schroeder, "Using Encryption for Authentication in Large Network of Computers," *Communications of the ACM*, Vol. 21, No. 12, December 1978, pp. 993 – 999.

[12] Roger M. Needham and Michael D. Schroeder, "Authentication Revisited," *Operating Systems Review*, Vol. 21, No. 1, January 1987, p. 7.

[13] B. Clifford Neuman and Stuart G. Stubblebine, "A Note on the Use of Timestamps as Nonces," *Operating Systems Review*, Vol. 27, No. 2, April 1993, pp. 10 – 14.

[14] Dave Otway and Owen Rees, "Efficient and Timely Mutual Authentication," *Operating Systems Review*, Vol. 21, No. 1, January 1987, pp. 8 – 10.

[15] J. G. Steiner, et al., "Kerberos: an Authentication Service for Open Network Systems," *Proceedings of the Winter 1988 Usenix Conferences*, Dallas, Texas, February 1988, pp. 191 – 201.

[16] Paul Syverson, "On Key Distribution Protocols for Repeated Authentication," *Operating Systems Review*, Vol. 27, No. 4, October 1993, pp. 24 – 30.

[17] Raphael Yahalom, appeared in *A Logic of Authentication*, DEC SRC Research Report 39, February 1990.