EVALUATING CRYPTOGRAPHIC PROTOCOLS

Alec F. Yasinsac and William A. Wulf

Abstract

Cryptographic Protocol (CP) analysis is a topic of intense research. Meadows describes four approaches for CP verification under investigation in [MEA92] and several authors have categorized protocols based on types of errors they are subject to [BIRD92], [SYV93a],[SYV93b]. This paper addresses the weakness injected into protocols when information is passed in the clear or encrypted only under the private key of a public/private key pair. We also propose a method for logically analyzing protocols based on action list analysis of valid and compromised protocol runs and of valid protocol runs interleaved with action lists of intruders conducting known attacks.

Section I. Introduction

In [NS78] Needham and Schroeder lay the foundation for research in the field now known as Cryptographic Protocol Verification (CPV). In that paper, the authors present cryptographic protocols for systems with authentication servers for shared and public key authentication and for what is now known as digital signatures. Those protocols effectively highlight many of the important issues in cryptographic protocols. They are used in practice, and in literature for purposes of illustration. In 1981, Denning and Sacco [DENN81] pointed out a subtle error in the Needham and Schroeder private key protocol, setting the stage for a flood of research aimed at examining cryptographic protocols.

In [BAN90] Burrows, Abadi, and Needham make a giant step forward in resolving the CPV problem. The Logic of Authentication (hereafter called BAN Logic) they propose has achieved widespread acceptance as the *de facto* standard system for CPV, and has been the target of "enhancements" and supplements by various authors [GNY90], [AT91], [SYV93a]. While BAN Logic is widely accepted, Nessett's paper [NESS90] demonstrated a weakness in the use of BAN Logic for protocol verification. This triggered broad and often spirited discussions regarding cryptographic protocol goals, environments, and other categorizations [BAN90b], [SYV91], [SNEK91]. Moreover, in much the same way as Denning and Sacco [DENN81] opened Pandora's box for examining cryptographic protocols, [NESS90] challenged the very mechanisms used to evaluate protocols and triggered intense research in suitable CPV mechanisms[BAN90b], [SYV91], [SNEK91]. As a result, CPV mechanisms are now subjected to the same level of scrutiny as the protocols they evaluate. Several fundamentally different mechanisms have evolved in the research.

In [MEA92], Meadows identifies the four major categories of mechanisms as:

1 - Use of specification and verification tools developed for software evaluation.
2 - Testing tools based on expert systems which evaluate different scenarios.
3 - Model the protocol requirements formally in an epistemic logic.

4 - Model the protocol formally based on the algebraic term-rewriting properties of cryptographic systems.

Examples of the first, second and fourth of these are compared in [KMJ93] and described in detail elsewhere in the literature. BAN Logic is an example of the third of these categories. Kemmerer proposes a method which spans the first and second categories in [KEM89]. The method we propose later in this paper spans the second and third categories.

In conjunction with attempts to address the CPV problem, there has been significant research into the nature of attacks on cryptographic protocols [BIRD92], [SYV93a],[SYV93b]. Bird et al. extend this research to offer guidelines for creating protocols which are not vulnerable to several of the known attacks. This line of research appears to be helpful in finding a mechanism suitable for effecting CPV.

While the efforts to establish a CPV methodology have contributed significantly to the understanding of cryptographic protocol complexity, no method has yet achieved widespread acceptance and use. In this paper, we propose a new approach to evaluating cryptographic protocols and highlight a weakness of the BAN Logic evaluation strategy. Section II of this paper addresses the difficulty that logics have dealing with time in evaluating cryptographic protocols. Section III discusses some attacks on protocols and suggests that the BAN authors should rethink their decision not to deal with cleartext messages in protocols. Section IV offers a new paradigm in evaluating cryptographic protocols. Section V summarizes and concludes the paper.

Section II. The Problem with Time

In [SNEK91] Snekkenes points out that BAN Logic is not capable of identifying weaknesses in annotated protocols caused by step permutations. The difficulty arises because BAN provides for only three discreet times in a protocol run [BAN90, p24]:

1 - Before the protocol run commences.
2 - While the protocol run is active.
3 - After the protocol run is over.

The annotation process is then conducted in three phases which correspond to the above times:

1 - Idealize the protocol and construct the assumptions (beliefs held before execution of the first protocol step)
2 - Apply the rules of inference and annotation rules to the protocol steps and formulae constructed from the first step then recursively using derived predicates.
3 - Identify the goals (beliefs to be held after the last protocol step) and determine if they are among the theses proved from step 2.

In the second phase, the analyzer may begin annotating with any desired protocol step. Annotation may begin with the last step, and conclude with the first. Since beliefs are derived as the result of applying annotation rules, the only formal means of enforcing sequencing on assignment of beliefs is in the application of individual annotation rules. Beliefs developed as postconditions

of one annotation may be thought of as evolving since they may become preconditions for other rules of inference, which are then used to derive other annotations. This describes the “evolution” of beliefs the authors of [BAN90] refer to. Unfortunately, this reflects logical evolution, i.e. deduction, rather than the temporal evolution which we desire to represent.

Snekkenes suggests finding a different collection of annotation rules in order to resolve this problem. In [GNY90] the authors set out a “BAN-like” set of constructs and annotation rules which contain weak temporal operators like “M not originated here” and “P once said X” in an attempt to include temporal sequencing power in the rules of inference. Unfortunately, the logic they propose received criticism for its complexity [SYV91] and has not achieved widespread acceptance.

We have not seen any mechanism which utilizes the “form” [i.e., sequence] of the actions of principals to temporally evolve beliefs as steps are executed. Evolution of beliefs from step to step is intuitive for cryptographic protocols. Principals in a protocol run actually acquire beliefs based on each action they take. A principal A may, for example, believe that “a message m can only be read by principal B” if A has just encrypted m under an encryption key shared only by A and B or A may acquire the belief that “m was sent by B” if A just decrypted m using her private key with B. The actions that are most likely to modify belief sets of principals are receiving and decrypting messages and comparing values recently received against values stored from previous sessions.

A related issue is how to handle the situation where conflicting beliefs arise. Burrows, et al. address this difficulty in [BAN89,p341], noting that false beliefs are likely to occur in protocol evaluation. So while party A may have a false belief about predicate P, certainly another (or maybe the same) party may simultaneously have the true belief about P. What is more, since principals gain information from each step in the protocol, it is reasonable to assume that principals belief about a proposition may change; Nessett demonstrates this in his protocol in [NESS90]. In that protocol, a user may rightfully believe that the key that was generated is in fact a “good” key before (temporally) the first message was sent. It is after the first message is sent that the key in fact becomes “bad”[1]. We will address the problem of compromising secrets during protocol runs in more detail in the next section.

The question of how to handle conflicting beliefs is far from resolved. Moser [MOS89] proposes a nonmonotonic logic based on the use of an “unless” operator which could be used to address this question. Syverson criticizes her logic [SYV91] as unnecessary and shows the equivalence of logics of knowledge and logics of belief in [SYV92]. At best, with Moser’s logic the system evaluator would be required to intuitively foresee the “unless” conditions when annotating the protocols, and provisions would have to be made to identify when conflicting beliefs actually occur.

Section III. Attacks on protocols

While Nessett showed that obviously flawed protocols can pass BAN Logic protocol evaluation, others have identified types of active attacks by intruders that protocols may be vulnerable to [BIRD92],[MOORE88],[SYV93a],[SYV93b]. In [BIRD92] the authors extend this evaluation to suggest guidelines for developing protocols that can reduce vulnerability of protocols to certain types of attacks. One of the key points they make is that different fields in each message should be cryp-

[1] This was pointed out to me in a mail message from Jon Millen.

tographically separated by Cipher Block Chaining encryption (cryptographically joining all bits in the message) of the concatenation of all the fields, or by sending a cryptographic checksum with each transmission. This is to prevent the attacker from accessing individual parts in transmitted messages and then controlling one field through another field, e.g., fooling one principal into generating a specific field that is needed in order to impersonate a user in another session. One of the by-products of this guideline is to suggest elimination of cleartext messages in cryptographic protocols. In fact, the attack Bird et al. document against the ISO protocol directly involves use of the messages passed in the clear. Though the vulnerability in this protocol results largely from the structure of the messages in the protocol, cleartext contributed to the weakness. We now describe how cleartext messages themselves present problems in cryptographic protocols.

In [NESS90] Nessett points out that BAN Logic does not have predicates to reflect the lack of secrecy of information passed in a protocol session. The authors of BAN indicate that information passed in the clear is not considered when idealizing the protocol because it can be forged [BAN89, p 330]. Unfortunately, though cleartext messages cannot add to the security of protocols, they can easily compromise that security if, for example, a cleartext message contains a key that is believed to be good by some participant in the current protocol run. This is effectively what Nessett does in [NESS90], passing a key encrypted under only the principal's private key of a public/private key pair, having virtually the same effect as passing the key in the clear. Cleartext transmission also plays a part in the attack Syverson describes on the Neuman-Stubblebine protocol in [SYV93b]. In that attack, the fact that the nonce is sent in the clear in the first message allows the intruder to compromise the protocol by later utilizing the nonce as a key.

In considering whether or not to use cleartext messages in protocols, one must consider performance implications. A stated reason for conducting BAN Logic evaluations in [BAN90] is to identify unnecessary encryption operations. It is also true that adding encryption operations also adds to the complexity of the protocol operations and possibly to the assumptions that must be made. The trade-off of performance for security must be judged by the user of the protocol. Since absolute confidence in the security of protocols cannot yet be achieved, the level of confidence added by eliminating cleartext messages in protocols is a subjective matter. Other factors such as availability of public key and centralized authentication servers also affect the decision to use or not use cleartext messages in a protocol. Nonetheless, if a protocol contains cleartext messages, their content must be included in any evaluation of its security.

Finally, many of the papers written on CPV have offered explication regarding goals of protocols, yet few discuss the goals and possible goals of intruders. Many intruder's goals are intuitive and reflect the antithesis of the protocol goals, e.g., attempting to masquerade as a principal in an authentication protocol. Other attacks may be more subtle, often reflecting the assumed, but not stated, goals of protocols, e.g. starting a spurious session against an authentication protocol. Some of the illustrated and more obvious goals of intruders are:

- Masquerade as a valid principal in a communication session with another valid principal and compromise the session key. This may be done with or without participation of the principal being impersonated and reference session may be left incomplete. The intruder may initiate the session or may "intrude" into a session begun by a valid principal.

- Begin a spurious session with or without the participation of the principal being impersonated though the session key is not compromised.
- Compromise the session key for a communication between two valid participants.
- Compromise the meaning of the messages in a session without compromising the key.
- Deny service to one or more valid participants without their immediate detection.

Though this list is not intended to be all inclusive, it gives a starting point for anticipating the actions of intruders in protocols.

Section IV. Cryptographic Protocol Step Interleaving

A. Specifying Cryptographic Protocols

Cryptographic protocols are routinely expressed in one of a number of pseudocode languages, where each protocol step represents a principal sending a message. These steps roughly take the form:

$$\mathbf{A \rightarrow B: \{X, Y\}^{kab}},$$

meaning principal A is sending to principal B the messages X and Y encrypted under some common encryption algorithm and under the common private key kab. Because they evolved for the purpose of specification, a common characteristic of these pseudocodes is that they represent the protocols in a high level of abstraction. As illustrated above, a single send operation represents numerous actions that the principals must take. While this is effective for protocol specification, it may result in difficulties in analzing them. For example, the receive action is implicit in the send operation. Now it is intuitively clear that when a principal A sends a message M to principal B, that B is expected to receive M. However, omitting the receive operation prevents us from expressing or considering actions that may occur ***between*** the originator's send and the destinations receive actions.

Similarly, the send operator encompasses the encryption and decryption actions. As a result, it is difficult to determine if the sending principal encrypted the message or if the principal is forwarding a message that was previously encrypted by another principal. In current practice, context is required to make this determination. Assigning responsibility for all actions to principals is greatly complicated by the broad spectrum of actions represented by the pseudocode send operation.

There are two other considerations that are more important to protocol analysis than specification. The first isthe need for mechanisms for specifying assumptions, such as preexisting private keys between participants. In order to mechanicall evaluate the success of the protocols, the assumptions must be expressed in a form suitable for this evaluation. The second is that most of these pseudocodes provide no decision operator. Conditions for continuation from step to step and for success of the protocol are either left to intuition or specified in a narrative accompanying the protocol.

Clearly, a more explicit language is needed to effectively evaluate cryptographic protocols. We

propose a language for that purpose, CPAL, in [YW93] which we will use here for describing protocols. CPAL specifically addresses the weaknesses described above. CPAL de-couples the actions of the pseudocode send operator by providing separate operators for sending (->), receiving (<-), encrypting (e[]k), and decrypting (d[]k) messages. This allows specific assignment of actions to principals which is accomplished by requiring that each statement be preceded by the identifier of the principal that is taking the action. Secure send (=>) and receive (<=) operators are used by principals to reflect assumptions of valid keys existing before the start of the protocol run. Finally, a conditional operator is provided to allow explicit description of the requirements for acceptance of messages.We will use CPAL to describe protocols for the duration of this paper, and will represent principal actions as statements in CPAL. More detailed descriptions of the language are found in Appendix A and in [YW93].

To interpret CPAL we use the model described in [AT91]. In that model, every message sent by a valid principal is intercepted by a powerful intruder. Upon intercepting a message, the intruder may select any of several courses of action, of which the interesting ones include:

- Forward the message to its intended destination as received,
- Save the message for later transmission,
- Modify the message and forward it as addressed,
- Combine parts of the message with other previous messages,
- Readdress all or parts of the message,
- Replace the message with its own or a previously recorded message.

A more formal description of the model is provided in Appendix B.

We believe the model we describe is complete in the respect that we can effectively represent any type of attack on a protocol. A normal run of a protocol is represented by the intruder forwarding all messages as intended by their originator. Attacks employing parallel or reference sessions are modelled by a single intruder or the intruder can masquerade as multiple principals and/or multiple intruders. Adopting this model gives a semantics to the language that is quite helpful in observing minor variations that may occur in a given protocol and facilitating evaluation of versions of protocols.

B. Protocol Actions

As we just described, it is common practice to specify protocols as an execution trace of actions by two or more principals. The actions are normally intermixed, with the principals “taking turns”. These protocols are expressed as though they are executed serially, on a single processor, when in fact they are intended for execution by separate processes, usually on different processors in remote locations.

An alternative way to view a protocol is as separate lists of actions for each principal, interleaved together to form an execution trace. The separate actions are synchronized in the execution by the blocking nature of the sending and receiving of messages between principals, so a principal cannot execute a receive operation until a send operation addressed to that principal containing the expected message has been enacted. Ordering of actions other than sending and receiving mes-

sages don not effect the meaning of the protocol. The only consideration is that the order of the original action lists must be maintained. Though there may be many different interleavings of the same action lists that meet these restrictions, their meanings will all be the same, as given in the following theorem:

> *Theorem 1*: Given two or more cryptographic protocol principal action lists, any interleaving of those action lists which preserves the order of the corresponding send and receive operations in the resulting trace has the same meaning as any other interleaving of the same action lists which also preserves the order of send and receive operations in the resulting trace.

The truth of this theorem follows intuitively from the nature of the environment upon which we base our evaluation. Each principal operates in an independent address space. If the meaning of a protocol is based on the states of the private memory of the principal's, the only way for a principal to impact the meaning of a protocol is by sending or receiving a message. Any other actions can only affect their own private memory. This means that the order of the interleavings of those actions cannot impact the meaning of the protocol.

The importance of these observations is to show that the combined semantics of the principal's action lists of a protocol are the same no matter what interleaving is considered. This fact allows us to construct traces using action lists other than those intended by the creator of the protocol. Specifically, we may desire to interleave action lists extracted from specified protocols with action lists derived from traces of known attacks [BIRD90, SYV93a]. These interleavings, which are nothing more than different versions of the protocol, can be mechanically evaluated in a standard method, such as BAN Logic. We believe the evaluation of these interleavings satisfies the need for the predicate suggested by Nessett [NESS90] that describes the secrecy of information passed in the protocol run. By evaluating the belief set of the intruder at the conclusion of the run, we can determine if any keys or other supposedly private information is within the belief set of the intruder (e.g. whether or not the intruder believes that kab is a good key). This is done without adding postulates to BAN Logic as suggested by Nessett. In this method, we extend the application of BAN Logic to evaluate the *security* of protocols against known attacks without supplementing or adding complexity to BAN Logic itself.
.
The purpose of this evaluation is not to demonstrate the complete security of a protocol, but to demonstrate it's security against specific known attacks. Because many attacks are known and mechanical evaluation is possible, many evaluations can be done. This offers a greater level of confidence than evaluation of only the specified protocol version with BAN Logic or other standard methods alone.

C. Evaluating Action Lists

As an additional benefit of this methodology, further useful information beyond formal proof of a protocol's security against a specific attack will be gained by conducting the evaluations of the different versions of the protocol. As with BAN Logic, the evaluations will point to weaknesses and inconsistencies that may have been otherwise unnoticed. We have found this to be true even when evaluating only a single principal's action list. Since an action list reflects all actions and

interactions of a principal in a protocol run, each action list can be evaluated using BAN Logic. We recorded such an evaluation of the originating principal's actions for the protocol given in Appendix C in Appendix D. In this evaluation, we found a subtle inconsistency in the evaluation of the protocol in [BAN89]. Our evaluation illustrates that principal A can only deduce her own belief in the validity of the session key, while the authors of [BAN89] suggest that A has the belief that B believes in the validity of the key. They claim A can have this belief because of the interaction of the last two messages of the protocol. Their conclusion may be based on their assumption that there is sufficient redundancy in each message to unambiguously decrypt each message. However, redundancy not withstanding, our evaluation shows that A has no mechanism to ensure that the value supposedly received from B was not simply a random number of the right size generated by a third party. In other words, neither the encrypted nor cleartext version of the last message transmitted by B is "verifiable" to A. This problem would be corrected by including B's identity along with N2 in B's last message, replacing it with:

B: ->A{B,N2}kab;

A could then verify that B's identifier was encrypted under kab and would then be justified in believing that B was aware of the key and had applied it to the last message.

Section V. Summary and Conclusions

In this paper we proposed a language for specifying and analyzing cryptographic protocols and described how to use this language in a new paradigm for conducting BAN Logic evaluation of protocols. We have used the language and paradigm to show an inconsistency in the BAN Logic evaluation of the Needham and Schroeder protocol, and have illustrated how research into attacks on protocols may be used in conjunction with this paradigm to extend the applicability of BAN Logic to evaluation of the security of protocols.

Evaluation of both protocol action lists and protocol interleavings adds to the ability of a cryptographic protocol designer to increase their confidence that a protocol meets its desired goals. Since no method now exists to ***guarantee*** that a protocol meets its goals, these steps are important, particularly so because they can be mechanized. In this light, our research continues as we pursue formal methods to improve confidence in cryptographic protocols.

Clearly, the next step is to automate a formal semantics for CPAL. We introduce such a semantics in [YW93], where the semantics of CPAL is described using the Weakest Precondition methodology of Dijkstra. Evaluation of the meaning of a protocol will be conducted in three steps:

1 - Specify the protocol in CPAL
2 - Mechanically construct the Verification Condition for the protocol
3 - Show that the assumptions of the protocol imply the verification condition

The definitions of CPAL statements will rely upon the model we described here and upon the inherent sequencing provided by the weakest precondition methodology. Of particular importance is the concept of private memory for each principal, so the meaning of a protocol to a principal can only be determined by what each principal sends and receives over the network.

It may also be possible to combine the mechanized methods we describe here with other methods described in the introduction to this paper in a workbench for cryptographic protocols. Translators from various pseudocode languages to CPAL and between CPAL and the language of Interrogator [MIL87] are of interest.

Bibliography

[AT91] Martin Abadi and Mark R. Tuttle, “A Semantics for a Logic of Authentication”, Tenth Annual ACM Symp on Princ of Dist Computing, Montreal, Canada, August, 1991

[BAN90] Burrows, M., Abadi, M., and Needham, R. M. “A Logic of Authentication”, ACM Transactions on Computer Systems, Vol. 8, No. 1, Feb 1990, pp. 18-36.

[BAN90b] Burrows, M., Abadi, M., and Needham, R. M., “Rejoinder to Nessett”, ACM Operating Systems Review, vol. 24, no. 2, April 1990, pp. 39-40

[BIRD92] Ray Bird, Inder Gopal, Amir Herzberg, Phil Janson, Shay Kutten, Refik Molva, and Moti Yung. “Systematic Design of Two-Party Authentication Protocols.” In Joan Fegenbaum, editor, Advances in Cryptology - CRYPTO ‘91, volume 576 of Lecture Notes in Comuter Science. Springer Verlag, Berlin, 1992

[CHEN90] Cheng, Pau-Chen and Gligor, Virgil D. “On the formal specification and verification of a Multiparty Session Protocol”. From 1990 IEEE Computer Society Symposium on Research in Security and Privacy, pp. 216-233

[DENN81] D. E. Denning and G. M. Sacco, “Timestamps in key distribution protocols,” Communications of the ACM, vol. 24, no. 8, Aug 1981, pp. 533-536

[DOL83] Dolev, D., and Yao, A.C. “On the security of public key protocols”. IEEE Trans. Inf. Theory IT-29, 2(Mar. 1983), pp. 198-208

[GLAS88] Janice I. Glasgow and Glenn H. MacEwen, “Reasoning About Knowledge in Multilevel secure Distributed Systems”, in Proceedings of the 1988 IEEE Syposium on Security and Privacy, Washington (IEEE), 1988, pp. 122-128

[GNY90] Gong, L., Needham, R., and Yahalom, R. “Reasoning about Belief in Cryptographic protocols”. From 1990 IEEE Computer Society Symposium on Research in Security and Privacy, pp. 234-248

[KEM89] R. A. Kemmerer, “Using Formal Methods to Analyze Encryption Protocols,” IEEE Journal on Selected Areas in Communications, vol. 7, mo. 4, pp. 448-457, May 1989

[KMJ93] R. Kemmerer, C. Meadows, and J. Millen, “Three Systems for Cryptographic Protocol Analysis”, To appear in The Journal of Cryptology

[LAMP91] B. Lampson, M. Abadi, M. Burrows, and E. Wobber, “Authentication in Distributed Systems: Theory and Practice”, ASM OS Review, Vol 25, No. 5, Special Issue, Proceedings of the 13th Symposium on Operating System Principles, 13-16 Oct 1991, pp 165-182

[MEAD89] Meadows, C., “Using Narrowing in the Analysis of Key Management Protocols”. From 1989 IEEE Computer Society Symposium on Research in Security and Privacy, pp. 138-147.

[MEAD91] Meadows, C., “A System for the Specification and Analysis of Key Management Protocols”. From 1991 IEEE Computer Society Symposium on Research in Security and Pri-

vacy, pp. 182-195.

[MEAD92] Meadows, C., “Applying Formal Methods to the Analysis of Key Management Protocols”, Journal of Computer Security, Vol. 1, No. 1, 1992

[MIL87] Millen, J.K., Clark, S. C., and Freedman, S. B. “The interrogator: Protocol security analysis”. IEEE Trans. Sofw. eng. SE-13, 2(Feb. 1987), pp. 274-288

[MOORE88] Judy H. Moore, “Protocol Failures in Cryptosystems”, Proceedings of the IEEE, Vol. 76, No. 5, May 1988

[MOS89] L. Moser, “A Logic of Knowledge and Belief for Reasoning about Computer Security” in Proceedings of the Computer Security Foundations Workshop II, Washington (IEEE), 1989, pp. 57-63

[NS78] Needham, R. M., and Schroeder, M. D. “Using encryption for authentication in large networks of computers”. Commun. ACM 21, 12 (Dec. 1978), pp. 993-999

[NS87] Needham, R.M. & Schroeder, M.D., “Authentication Revisited”, ACM Operating Systems Review, Vol. 21, No. 1, January 1987.

[NESS89] D. M. Nessett, “Layering Central Authentication on Existing Distributed System Terminal Services”, From 1989 IEEE Computer Society Symposium on Security and Privacy, pp. 290-299.

[NESS90] D. Nessett, “A Critique of the Burrows, Abadi, and Needham Logic”, ACM Operating Systems Review, vol. 24, no. 2, April 1990, pp. 35-38

[OTWY87] Otwy, D., and Rees, O. “Efficient and timely mutual authentication”. Operating Systems Review 21, 1(Jan. 1987), pp. 8-10

[SNEK91] Snekkenes, E., “Exploring the BAN Approach to Protocol Analysis”. >From 1991 IEEE Computer Society Symposium on Research in Security and Privacy, pp. 171-181.

[SYV91] Syverson, P., “The Use of Logic in the Analysis of Cryptographic Protocol”., From 1991 IEEE Computer Society Symposium on Research in Security and Privacy, 156-170

[SYV92] Syverson, P., “Knowledge, Belief, and Semantics in the Analysis of Cryptographic Protocols”, Journal of Computer Security 1 (1992), pp 317-334

[SYV93a] Syverson, P., “Adding Time to a Logic of Authentication”, in Proceedings of the First ACM Conference on Computer and Communications Security (Fairfax VA, Nov 3-5).

[SYV93b] Syverson, P., “On Key Distribution Protocols for Repeated Authentication” To appear in Operating Systems Review vol 27, no 4, October 1993, pp. 24-30.

[RANG88] P. Venkat Rangan, “An Axiomatic Basis for Trust in Distributed Systems”, in Proceedings of the 1988 IEEE Symposium on Security and Privacy, pp. 204-211, IEEE Computer Society Press, Washington, DC, 1988

[YW93] Alec Yasinsac and Wm. A. Wulf, “A Formal Semantics for Evaluating Cryptographic Protocols”, Unpublished manuscript, August 1993

APPENDIX A - Description of CPAL

The purpose of this language is to provide protocol developers a language that is sufficiently rich to express cryptographic protocols, yet small enough to allow creation of an easily understood, unambiguous formal semantic for the language. To make the language expressive, we allow arbitrary selection of identifier names for principals, messages, nonces, and keys. The operators are designed to allow expression of actions of principals in a protocol run by providing for sending, receiving, encrypting, and decrypting of messages; generating keys, nonces, and timestamps; computing functions; and making decisions using an IFTHENELSE structure. This language allows explicit expression of assumptions by providing send and receive off-line operators so *no* prior knowledge by participants need be assumed. There is no iteration constructor in CPAL. As a brief description of the language, the following notes are offered:

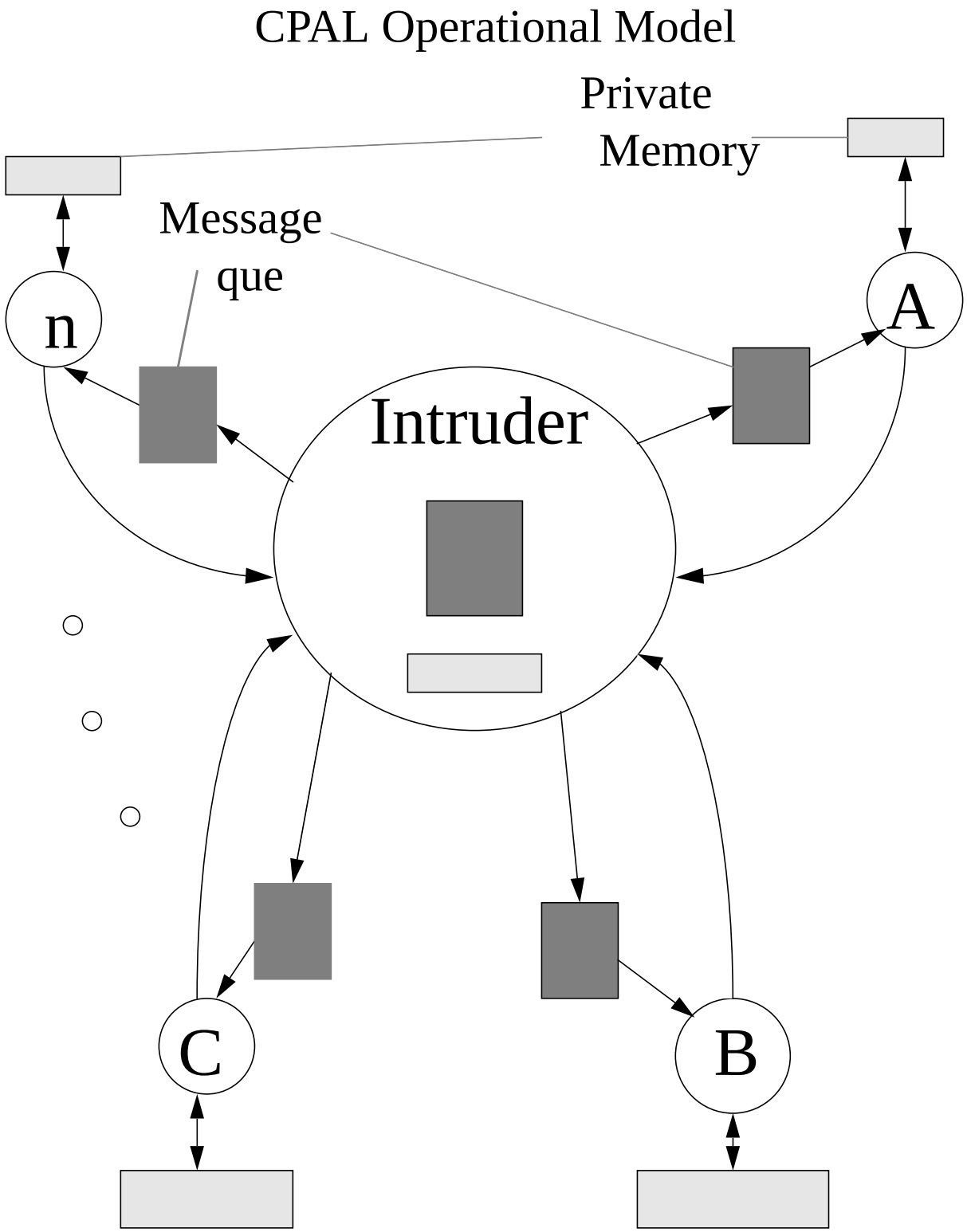
Since all interactions between participants must be explicit. Each protocol step will be labeled with the ID of the user taking the action.

Messages are sent/received via the unsecure send (->) and unsecure receive (<-) operators. The intended destination for a sent message is appended to the right of the send operator. For the receive operation, no identifier is allowed to represent network provided identification of the sender for unsecurely received messages. The identification of the sender must be determined by the cryptographic method. The secure send (=>) and receive (<=) operators are included to depict assumptions and for use by an all powerful intruder. The format for secure send is the same as send, but the secure receive will have the sender and intended receiver appended to the operator.

The format for encryption/decryption is an ad hoc standard of preceding curly braces with ‘e’ or ‘d’ with the message identifier within the braces. The identifier for the encryption/ decryption key follows the right curly brace. Keys can be represented by an arbitrary identifier. For purposes of convention, we allow the characters ‘+’ and ‘-’ to suffix identifiers which could be used to represent public and private keys. For example, user A’s public/private key pair might be ka+ and ka-.

Concatenation of values is signified by a comma. Statements are separated by semicolons.
‘:=’ is the assignment operator. Simple and compound assignment is allowed.
‘==’ is the comparison operator.

```
A: ->B(M1);                          [User A sends message M1 addressed to user B]
B: <-(M1);                           [User B receives message M1, originator ID must be in M1]
A: =>B(M2);                          [User A sends message M2 to user B off the net]
I: <=A->B(M1);                       [Intruder intercepts message M2 sent by A addressed to B]
A: M4 := e[M3]kab;                   [A encrypts M3 under A & B's private key kab into M4]
A: M5 := <X,Y,Z>;                    [Assignment of the concatenated values X,Y, and Z]
A: (Q,R,S) := M6;                    [Compound assignment of the structured value M6]
A: If (V1 == V2) then S1 else S2;    [If-then-else statement]
A: N1 := new;                        [N1 is a new nonce, ie. A has jurisdiction over N1]
A: Y :=  f(X);                       [Compute a function of X, store in Y]
```


CPAL Operational Model
Private
Memory
Message
que
n
A
Intruder
C
B

APPENDIX B

Operational Model for CPAL

Principal. A principal is a process. Principals are assumed to know protocol steps. Principals are identified by a bitstream (name) unique to that principal. Resources available to principals are:

- Substantial, but not unlimited, processing power,
- Substantial, but not unlimited, private memory
- Transmission medium for sending messages bit by bit
- Message que for receiving and temporarily storing messages

Intruder. *The intruder is a principal* that receives all messages sent by all other principals.

Private memory. No other principal can detect the contents of this memory. In this large memory space, principals may store, modify, and retrieve values represented as bit streams such as:

- Independently created messages,
- Copies of messages received from another principal,
- Edited messages
- State information related to the protocol
- Identifiers of other principals
- Any other bit stream the principal desires to store

Concatenate. The concatenation operator is the [infixed] comma, e.g. M1,M2. Concatenation is performed on values.The result of concatenating values is to combine two or more values into one value from which the original values may be recreated.

Encrypt. Encrypt is performed on a value, though the value encrypted may be the concatenation of two or more values. The result of encrypting a value is to produce another value.

Decrypt. The result of encrypting a value is to produce another value. The resulting value may be the concatenation of two or more values.

Assignment. Simple assignment, with a single destination (left side) identifier, is nondestructive, pass by value assignment. If the source (right side) also contains a single identifier, the source value is copied into the memory location identified by the destination identifier. If the source contains multiple identifiers separated by the concatenation operator, the values are concatenated and the resulting value is stored in the destination memory location.

For compound assignment (two or more destination identifiers), the assignment operation must “unconcatenate” or separate concatenated values. Each concatenated source value is copied into the memory location designated by the destination identifiers. Source values are bound to destination names by order, with the first value in the concatenation copied into the location designated by the first (leftmost) destination identifier. The number of concatenated values must equal the number of destination identifiers or the operation will fail.

Message. A message is a value represented by a bitstream. Messages may be encrypted, decrypted and/or concatenated recursively.

Send. Principals may (nondestructively) transmit messages stored in their private memory to other principals. A principal's identifier is synonymous with their address. The result of a normal send operation is that the message will be appended to the message que of the intruder and the corresponding que counter will be incremented.The result of a secure send operation is that the message will be appended to the message que of the appropriate principal and the corresponding que counter will be incremented. Concatenated messages are transmitted, and will arrive in the appropriate que, as a single message.

Receive. Principals may receive messages which have been appended to their message que. Receive is a blocking operation. If a message is in the que when a receive operation is executed, the message is copied to the principal's private memory and deleted from the que, and the que counter is decremented. If the message que counter is zero when a receive is executed, block will occur until a message is appended to the que.

Transmission medium. Principals other than the intruder send all outgoing messages, bit by bit, via a single output channel connected directly and only to the Intruder's message que. No assumptions are made regarding the privacy of messages on message ques.

Message Que. Principals can only receive messages that are stored on their [FIFO] message que. The intruder can place messages on other principals message que at any time. Other principals may update queues using the secure send operations Valid que operations are push and pop. The push operation will append a message to the que and increment the message counter, while the pop operation will copy the next message to be read to the private memory of the principal, delete the entry from the que and decrement the message counter.

APPENDIX C

Needham, R. M., and Schroeder Private Key Protocol

```
S: new(f);
S: =>A,B(f);
                A: <=S(f);
                                        B: <=S(f);
S: kas := new();
S: =>A(kas);
                A: <=S(kas);
S: new(kbs);
S: =>B(kbs);
                                        B: <=S(kbs);
                A: N1 := new();
                A: ->S(A,B,N1);
                                                        I: <=S-B(A,B,N1);
                                                        I: =>B(A,B,N1);
S: <-(Srcid,Destid,N1);
S: kab := new();
S: ->Srcid(e[N1,Destid,kab,e[kab,Srcid]kbs]kas);
                                                        I: <=S-A(e[N1,Destid,kab,e[kab,Sr-
                                                                cid]kbs]kas);
                                                        I: =>A(e[N1,Destid,kab,e[kab,Sr-
                                                                cid]kbs]kas);
                A: <-(Msg1);
                A: (N1',Destid,kab,ticket) := d[Msg1]kas;
                A: if (N1' == N1) then {accept(kab);   ->B(ticket);}
                   else reject;
                                                        I: <=A-B(ticket);
                                                        I: =>B(ticket);
                                        B: <-(ticket);
                                        B: (kab,Srcid) := d[ticket]ksb;
                                        B: N2 := new();
                                        B: ->Srcid(e[N2]kab);
                                                        I: <=B-A(e[N2]kab);
                                                        I: =>A(e[N2]kab);
                A: <-(Msg2);
                A: N2 := d[Msg2]kab;
                A: ->B(e[f(N2)]kab;
                                        B: <-(Msg3)
                                        B: N3 := d[Msg3}]kab;
                                        B: if (N3 == f(N2)) then accept; else reject;
```

Action lists for Needham, R. M., and Schroeder Private Key Protocol

```
S: new(f);
S: =>A,B(f);
S: new(kas);
S: =>A(kas);
S: new(kbs);
S: =>B(kbs);
S: <-(Srcid,Destid,N1);
S: kab := new();
S: ->Srcid(e[N1,Destid,kab,e[kab,Srcid]kbs]kas);
                A: <=S(f);
                A: <=S(kas);
                A: N1 := new();
                A: ->S(A,B,N1);
                A: <-(Msg1);
                A: (N1',Destid,kab,ticket) := d[Msg1]kas;
                A: if (N1' == N1) then accept(kab);   ->B(ticket);
                   else reject;
                A: <-(Msg2);
                A: N2 := d[Msg2]kab;
                A: ->B(e[f(N2)]kab;
                                B: <=S(f);
                                B: <=S(kbs);
                                B: <-(ticket);
                                B: (kab,Srcid) := d[ticket]ksb;
                                B: N2 := new();
                                B: ->Srcid(e[N2]kab );
                                B: <-(Msg3)
                                B: N3 := d[Msg3]kab;
                                B: if (N3 == f(N2)) then accept; else reject;
                                                I: <=S-B(A,B,N1);
                                            I: =>B(A,B,N1);
                                            I: <=S-A(e[N1,Destid,kab,e[kab,Sr-
                                                  cid]kbs]kas);
                                            I: =>A(e[N1,Destid,kab,e[kab,Sr-
                                                  cid]kbs]kas);
                                            I: <=A-B(ticket);
                                            I: =>B(ticket);
                                            I: <=B-A(e[N2]kab);
                                            I: =>A(e[N2]kab);
```

APPENDIX D

BAN Logic Evaluation of the N&S Private Key Protocol Originator Actions

```
A: <=S(f);
A: <=S(kas);
A: N1 := new();
A: ->S(A,B,N1);
A: <-(Msg1);
A: (N'1,Destid,kab,ticket) := d[Msg1]kas;
A: if (N'1 == N1) then {assert("kab is a good key");
                        ->B(ticket);}
   else reject;
A: <-(Msg2);
A: N2 := d[Msg2]kab;
A: ->B(e[f(N2)]kab;
```

The Idealized protocol	The annotations	#
A believes $A \overset{kas}{\leftrightarrow} S$;		
A believes S controls $(A \overset{kab}{\leftrightarrow} B)$;		
A: -> S(N1);		
	I sees N1	
A: <-(Msg1);		
	I sees Msg1	
	A sees Msg1	
A: $(N'1, A \overset{kab}{\leftrightarrow} B, [A \overset{kab}{\leftrightarrow} B]kbs) := d[msg1]kas$;		
	A sees (N'1, kab, {A$\overset{kab}{\leftrightarrow}$B}kbs)	
A: if (N'1 == N1) then A believes #msg1;		
A believes $(A \overset{kab}{\leftrightarrow} B)$;		
->B($[A \overset{kab}{\leftrightarrow} B]kbs$);		
	I sees {A$\overset{kab}{\leftrightarrow}$B}kbs	
<-(Msg2);		
	A sees Msg2	
N2 := d[Msg2]kab;		
	A sees N2 -- A cannot verify that the N2 she received is the same N2 that B sent.	
	A believes B believes A$\overset{kab}{\leftrightarrow}$B ?-no	
->B(e[f(N2)]kab;		
else skip;		