Verification and Modelling of Authentication Protocols[1]

Ralf C. Hauser

Institut für Informatik, Universität Zürich, Winterthurerstr. 190
8057 Zürich, Switzerland, hauser@ifi.unizh.ch

E. Stewart Lee

Department of Computer Science, University of Toronto, 10, King's College Road
Toronto, M5S 1A4, Ontario, Canada, stew@hub.utoronto.ca

Abstract. With the emergence of numerous distributed services, the importance of electronic authentication in networks is rapidly increasing. Many authentication protocols have been proposed and discussed. Burrows, Abadi and Needham [BAN1] created a logic of authentication to formally analyze authentication protocols. This BAN–logic has been subject to critique and several extensions have been suggested. Nonetheless, due to its straightforward design and its ease–of–use, it attracts the attention of current research. In this paper, an authentication logic is proposed which is built closely after the BAN–logic. It addresses answers to important criticisms of BAN like the non–disclosure problem, and avoids some newly discovered weaknesses of BAN, e.g. with respect to freshness. It also does not require any idealization which is a major hurdle to the correct usage of BAN. This extended BAN–logic is instrumented as a verification tool which also allows for modelling the different protocol participants as finite state machines. Also, actions of intruders, consequences of such intrusions, and the respective counter–measures can be modelled and simulated.

1 Overview

After outlining previous work, some remarks on the notion of authentication follow. Then, the proposed logic is described and it is explained to what extent the logic covers the given definitions. This logic's basic concepts and assumptions of principals, protocols and encryption schemes are introduced and included in a Prolog tool. Last, the deployment of Prolog which assures the logical soundness of the tool is explained.The developed tool is based on two building blocks which are exploited with strong interdependencies:

- an analytical protocol–verification part employing the new authentication logic and
- a simulation and modelling part based on finite state machine definitions of the participating principals.

These two parts facilitate exploratory design and verification of authentication protocols especially also through the modelling and simulation.

1. Most of the work for this paper was done for a thesis for the degree of a Master of Science in Computer Science in the University of Toronto.

Y. Deswarte et al. (eds.), *Computer Security - ESORICS 92*

2. Previous Work

The new logic is strongly based on the BAN logic of authentication. It has similarities with another BAN–extension of Gong, Needham and Yahalom [2], although the new logic maintains the straight–forwardness of BAN and draws significantly different conclusions than GNY.

2.1. BAN: Authentication Protocols and a Logic to prove them

The authentication protocol Needham and Schroeder proposed in their paper "Using Encryption for Authentication in Large Networks of Computers" [3] 1978 has experienced widespread attention and is the basis for several implementations. Under the assumption that the employed encryption is strong, mutual trust in the communicating principals' identity can be established by exchanging an authenticating secret over a possibly untrusted network. 1981, Denning and Sacco [4] pointed out the flaw of the protocol and in 1983 Bauer et al. [5] pointed out even more drastic consequences of the same flaw. The need for a systematic analysis tool was first addressed in 1989 when "A Logic of Authentication" was published [1]. This BAN–logic is applied as follows:

> Assumptions about the beliefs of an initial state are transformed by a small set of logical postulates to beliefs in a final state. This final state's conclusions must at least include each principal's belief in a shared session key or the partner's public key. Within this chain of applications of the logical postulates, each message of the protocol monotonically leads to a new, intermediate state of beliefs.

There are four main constructs:

$P \mid\equiv X$	P *believes* in X, or P would be entitled to believe X.
$P \mid\sim X$	P *once said* X. The principal P at some time sent a message containing the statement X which was correctly authenticated.
$P \mid\Rightarrow X$	P has *jurisdiction* over X. The principal P is administrating X. X is for example an encryption key that must be created with some care. Often, the principals trust a server to do so as an initial assumption.
$P \lhd X$	P *sees* X. Someone has sent a message containing X to P. P is able to read X and repeat it in other messages.

The *freshness* attribute and the *goodness of* shared and public/private *keys* are further issues in this analysis.

The main activities of the different principals are creating nonces, keys/secrets/passwords at various times, and encrypting and decrypting messages using and containing these items. The flow of these actions is strictly defined by protocols and encryption schemes assigning different competencies and capabilities to the principals. BAN has formalized the goals of authentication and those capabilities. The application of its postulates allows to automatically check the viability of the distribution of competencies and items during a protocol run. The three main logical postulates which advance each principal's beliefs after every received message are:

Jurisdiction: If P believes that Q has jurisdiction over X, then P believes X if P believes Q to believe X.

Nonce–verification: If X is fresh and the sender Q once said it, then P can believe that Q still believes it.

Message–Meaning: If P believes in the own shared key with Q and sees an item X encrypted with this key, P believes that Q once said X.

This logic system is defensive, because "... we allow for the possibility of hostile intruders ...". It is, however, not totally defensive because (a) " ... there is no attempt to deal with the authentication of an untrustworthy principal" and (b) "nor to detect weaknesses of encryption schemes ...". These restrictions are motivated as "Our goal however is not to provide a logic that would explain every authentication method, but rather a logic that would explain most of the central concepts of authentication. ... It is hoped that protocol designers will adapt it to suit their specific needs" [1] p.2.

The BAN analysis relies on an idealization procedure which is awkward to use because its rules are not transparent.

2.2. GNY: A successor of BAN

Under the title "Reasoning about Belief in Cryptographic Protocols", an extension of the BAN has been published by Needham et al. [2].

It introduces several new constructs like a differentiation between "being told" and "once conveyed". The first step of the analysis is a protocol parser which reverts this differentiation performing a "possession consistency check" and possibly adds the "not–originated–here" attribute. The labelling with this attribute requires some kind of global or common knowledge which the logic of the proposed tool never assumes.

"Recognizability" is another construct of GNY which seems unnecessary for the goals they achieve. It may gain importance when formally considering verifiable plaintext and poorly chosen keys [6] or the integrity of exchanged data (although the latter is often provided by the communications subnet and the continuation of a protocol with erroneous items normally ends with rejected challenges).

Their concept of the separation of a principal's state between a set of beliefs and a set of visible possessions is adopted in our present tool, but deployed differently.

They operate with a sizable set of logical postulates. Furthermore, they introduce the concept of "message extensions" which are preconditions for items to be conveyed. This leads to the conclusion that if principals convey a message depending upon such an extension, they also believe the extension. A principal receiving such a message ought to have no belief in other principals to do something beyond what is specified in the initial beliefs. Therefore, we refrain from including this feature in the analysis part of the tool and delegate this functionality aspect to the state machine simulation part of the new tool.

2.3. Tools: Kemmerer's usage of the Inatest System, the Interrogator

Kemmerer's work [7] is guided by similar goals as ours, but it uses its own peculiar set of axioms and rules instead of a widely known formalism like the BAN logic. Furthermore, its analysis doesn't achieve verification results of a generality similar to BAN. It only automates a sophisticated analysis of a specific protocol failure hypothesis given by its human user.

Conversely, Millen et al's Interrogator [8] exhaustively tests the protocols in order to uncover a flaw. Similarly to our tool, it uses Prolog to perform its actions and its "penetrator" and "network buffer"/"global state" concepts might be mapped to what we call an intruder and to the state machines of the simulation module. However, the new tool does not exclusively rely on exhaustive techniques to produce its analysis results. They are only useful add–ons to a formal proof with a BAN–type authentication logic.

Furthermore, the development of our tool could start from some existing BAN related Prolog code by R. Soper [9].

3. Authentication

In order to establish a proper base of the new logic the notion of authentication has to be discussed first.

Definition:

Authentication is a recent verification of the identity of a principal.

A principal is a participating subject of the network, e.g. represented by an interconnected host. Authentication has two components: identification and freshness.

Identity

A principal's identity can be proven with knowledge, with possessions, or with personal attributes and more than one of them must be employed to establish a high assurance of correct identification.

Knowledge is normally favoured in computer systems because it doesn't require expensive special–purpose hardware like fingerprint–readers etc., nor specific physical objects carried by users. This knowledge has to be kept secret to be valid as a proof of identity.

Even though it is isn't relevant for the proposed logic, authentication can be considered as a uni– or bi–directed activity which leads to notions like "authenticated datagram" in the sense of proven origin without the freshness quality.

Freshness

In distributed systems, reliable global time services cannot be assumed. Thus, timestamps are a problematic, or at least expensive, concept as they raise the need for protocols to provide secure time services. Therefore, random strings called *nonces* [3] are introduced. They are created for the purpose of demonstrating freshness.

A standardized sequence of steps normally including nonces provides the proof of freshness. This sequence of steps is called "Authentication Protocol". The employed logic calls a protocol a good authentication protocol

if information can be transmitted with such a ***recent*** *proof of identity.*

This definition does not require that a direct contact between the authenticating principals with any exchanged information must take place, this proof can be delegated to password servers respectively key distribution centers.

In practice, authentication rarely takes place as an isolated action in distributed systems. Therefore, a subsequent information exchange is assumed in Kerberos [10] and other authentication services. This implies after a successful authentication the existence of some confidentiality/integrity service protecting the entire conversation which is assumed to be based on symmetric, secret *session keys*. With the appropriate adaptation of the goals, the tool can also be used for the verification of other cryptographic protocols like e.g. key distribution protocols based on public keys.

4. The New Authentication Logic

The purpose of the new logic is like in BAN to provide a basic tool for protocol designers and to ease the illustration of protocols. Special care must be taken in specifying the variety of protocols the tool can be applied to and the resulting conclusions. This materializes in focusing especially on the initial assumptions and the idealization which in a way hook up the mechanism of the logic with a 'network–reality'.

To suit these purposes the mechanisms and the 'embedding' of the logic are explained as follows:

- What are the principals and what qualities make them distinguishable ?
- How is authentication formalized in the logic ?
- How are initial beliefs and protocol runs encoded ?
- How are the logical postulates adapted to the new model ?
- What are the "universal assumptions" of the model ?

As these 5 aspects have strong inter–dependencies, often several options for how to integrate different qualities into a protocol's system definition are possible.

4.1. The Principals – default participants in a realistic network model

The process of authentication encompasses at least the following different kinds of principals:

Regular or **Private hosts prh**
Protocols authenticate prhs to password servers/key distribution centers (ps/kdc) or even more frequently two different prhs to each other with the support of a ps/kdc. While the connections are considered unreliable or even under the control of the hostile intruder, the involved hosts are considered as trustworthy.

Prhs are either clients of network services called *user login hosts* or *application servers* providing these services. This functionality difference sometimes causes asymmetric initial assumptions because application servers do not need to be physically accessible to the public like the user login hosts.

The prhs are capable of performing strong encryption authenticating themselves to the ps/kdc. They are either trusted to store a well chosen key or secret between the sessions or they are fed with a weak key (password) by a user at the beginning of each session. In some cases they only must be able to create strong random numbers [6] as a third option.

The password server/key distribution center ps/kdc
Often, in a network there is at least one host with additional features:

- it is able to create good, i.e., well chosen, keys and secrets
- it is normally the only principal which is trusted to correctly 'jurisdict' secrets which are not its own.
- it maintains a database containing authenticating secrets for every valid prh of the network
- it is the network component deserving the highest level of trust

It mediates communication between n mutually distrusting prhs, allowing them to acquire pairwise session keys so as to reduce the number of required permanently stored keys from $O(n^2)$ to $< O(n)$. Furthermore, the usage of session keys limits the damage of disclosure to one session.
Because a ps/kdc is an inherently centralized concept, all the problems of availability/response–time versus integrity/redundancy will be encountered in practice. We are aware of this problem which can be modelled with the new tool, but we do not provide alternatives to avoid it in the current paper.
The ps/kdc can act as a network security authority. It has the so–called "competency" and "honesty" [2]. The trust in these qualities must be stated explicitly because honesty cannot be assumed as shown next (e.g. the prh believes that kdc controls good session keys; the jurisdiction rule requests belief in the other's jurisdiction legitimacy).

The hostile intruder

We summarize all possible hostile actions in a network under a hypothetical intruder. All protocols must be evaluated against this common or general intruder. It must be assumed to:

- perform every malign action except those excluded by explicit assumption,
- have complete control over the network resources,
- have hard– and software power of the most modern available technology at their disposal,
- neither forget nor miss the slightest information which might be disclosed by negligence,
- have possession of all public keys in the system (not all directory services presently require authentication).

Since an intruder may wiretap the communication lines, every message must be assumed to be a broadcast.

The insider

The insider is a special case of an intruder. Towards the ps/kdc, it is a normally legitimated principal but at the same time, it is acting as a intruder. An insider is one example of an "untrustworthy principal" which is explicitly excluded by BAN. GNY describes it as follows: "The ... assumption we require is, that principals do not reveal their secrets". Therefore, every principal is potentially an insider and good protocols have to take this into consideration (e.g. with session keys etc.). The notion of *insider–attack* might be defined as follows:

> A successful insider attack allows a principal who is trusted by the password server to obtain an advantage not available to the general intruder by employing parts of the trusted interactions without being identified as a security threat.

Other untrustworthy entities with coordinated actions among groups of untrustworthy cooperators (similar to tracker mechanisms for data bases) can be imagined. Coping with insiders remains a widely open question and only the protection against unidentified keys has been integrated in the current tool.

4.2. Formalization of *Authentication* in the Logic

The logic evaluates "good" protocols with two kinds of information to be transmitted with a fresh proof of identity:

- *Public* information as in the CCITT X.509 Protocol.
- *Secret* information as in all the other BAN examples.

In the first case, successful authentication has taken place, if the receiver is entitled to believe that the sender believed recently in its own statement (narrower definition than in section 3).

In all the other cases, BAN considers it sufficient if both parties believe in a key, be it a secret symmetric session–key or the partner's public key. As in Prolog, these conditions are called goals in BAN. The BAN goals are extended with respect to disclosure as follows:

Basic Non–Disclosure

Nesset states [11]: "The security of a protocol rests on two complementary properties. First it must distribute information to a subset of principals so that some predicate defined over the population obtains... However, the second property is just as important. The protocol must distribute information in such a way that another subset of the population is denied access to it." Although according to BAN, this non–disclosure property is conforming to a secondary purpose of most authentication protocols, namely establish a secure channel through a session key, it does not follow the strict definition of authentication.

To suit also these pragmatic purposes the new logic requires non–disclosure of certain items for a successful protocol verification. According to the full knowledge rule [12], everything is assumed to be publicly known by default. Thus, everything that must be kept secret, must be listed explicitly as "needs to remain secret" (also this session key itself), except for the conversation protected with the fresh session key,. The non–disclosure of all these items is the requirement of the verification proof.

To enable this check, the *unencrypted messages* cannot be neglected anymore (as in BAN).

Future improvements of the new logic might choose a more powerful notion of authentication closer to the definition of section 3. There might be different approaches to include the non–disclosure property other than through an extra principal, i.e. this common intruder.

4.3. Encoding of Initial Beliefs and Protocol Steps (Idealization)

The section about "Idealized protocols" of BAN discusses mandatory idealization steps and refers to guidelines on how to perform them. But these remain vague: "The idealized form of each message cannot be determined by looking at a single protocol step by itself. Only knowledge of the entire protocol can determine the essential logical contents of the message ...".

The proposed logic avoids the risk of introducing new errors by avoiding idealization steps. The protocol designer has the option either to model his protocol with a state machine as described later or to translate the messages into a Prolog clause and establish the necessary set of initial beliefs. Both options are some kind of encoding, but do not add implicit assumptions and information like BAN.

The set of initial assumptions is almost identical across all protocols and encompasses the following:

- Each principal's beliefs in the permanent key authenticating it to the ps/kdc. The distribution of these keys is not considered in this analysis and takes before (and possibly place out–of–band).
- The belief in an authority to create ('jurisdict'/control) good session keys, which is normally the ps/kdc.

– All principals create their own good nonces. As there is a dominance relation between three of the main BAN constructs, all principals believe what they create. Therefore everyone believes in one's own nonces. Furthermore, possession through seeing is a mainly functional precondition to believe something, therefore all principals see their own nonces.

The analysis of the differences among initial assumptions of protocols might be a distinction leading to a more sophisticated classification of protocols.

4.4. Adapted and Enhanced Logical Postulates

In this section two issues are discussed:

1) The proper separation of belief and visibility.
2) An improvement of the freshness rules.

4.1.1. Visibility independent of beliefs

The proposed system encompasses two interdependent worlds: the world of beliefs and the world of visibility. Conversely to BAN[1] p. 7, p. 32, the "seeing world" can exist without interconnections to beliefs. To see the content Y of a message, no belief in the respective decryption key K_X is necessary, simply seeing K_X and the entire message is sufficient.

$$\frac{P \lhd \{Y\}K_X, P \lhd K_X}{P \lhd Y}$$

Without the BAN idealization, the goals of the second order (P believes that Q believes K_{PQ} to be a good key) can not be deduced. In CSRI [9], this problem is solved by extending the message–meaning postulate by deducing that a key is "once said" when it is correctly applied. This is necessary since if $P \models K_{PQ}$, there is no assurance that Q ever saw K_{PQ}. Therefore, seeing Y which must have been encrypted by Q is the first proof of that.

$$\frac{P \lhd \{Y\}K_{PQ}, P \models K_{PQ}}{P \models Q \sim K_{PQ}}$$

The following kinds of encryption schemes are presently supported by the tool:

– Schemes with symmetric keys like DES [13].
– Schemes with asymmetric (public/private) keys like RSA [14].
– Schemes XORing with stream ciphers like the random cipher in the protocol against poorly chosen keys [6].

4.4.2. Freshness rules revisited

Freshness can have two characteristics. It can mean:

1. a fresh *creation* of something #c

or

2. a fresh *use* of something #u

This leads to confusion in the BAN and GNY logic because the definition of the freshness operator only refers to "creation", but the rules employing it work with the implicit notion of both characteristics. A simple example is a message from a kdc/pw–server to a principal containing a timestamp and the public key of a third principal. This public key is erroneously considered as fresh by the conclusion of the freshness rule with its logical OR–semantics. A more precise statement of the property this public key achieves is that "it bears the same authenticity as the timestamp it is sent with". This lack of distinction between fresh creation and fresh use can lead to flawed conclusions:

> E.g. if there are two messages both containing the same string "dummy" and the password server in the past said {dummy, K_X}K_{ps-prh} and recently said {dummy, timestamp(ps)} K_{ps-prh}, prh would believe K_X to be fresh by transitively deriving this freshness from dummy, although this could be a replay by a third party (dummy gained its freshness from timestamp(ps)).

Therefore we adapt the BAN freshness rule as follows:

Freshness can only be transferred among items of a message if the message is protected with a key believed to be good. Furthermore, freshness can only be transferred from a "freshcreated" item to another item which only achieves the "freshused" status.

$$\frac{P \models \#c(X),\ P \models \text{Key } K,\ P \lhd \{X,Y\}K}{P \models \#u(Y)}$$

Furthermore, if an encrypted message contains a freshcreated item, the employed key can be considered to be freshused.

$$\frac{P \models \#c(X),\ P \models \text{Key } K,\ P \lhd \{X\}K}{P \models \#u(K)}$$

Also, as a special case, a freshused session key can transfer this quality to the items encrypted with it. This requires 1) total trustworthiness of the ps/kdc which knows this key as a third party and 2) assurance that the key is only a session–key (which implies that it is somehow freshcreated at least in this session).

$$\frac{P \models \#u(\textit{Session}\text{key } k),\ P \models \textit{Session}\text{key } k,\ P \lhd \{X\}k}{P \models \#u(X)}$$

The receiving parties normally believe timestamps to be freshcreated.

4.5. Universal Assumptions

In this section, the universal assumptions which are not mentioned for every protocol verification and some working assumptions are explicitly presented. There are some congruencies with BAN:

- We allow for the possibility of *hostile intruders*.
- We do not provide complete protection against untrustworthy principals, i.e. *insiders*.
- "nor to detect weaknesses of encryption schemes ...".

Further assumptions are:

- Secrecy on the level of the traffic analysis problem and denial of service attacks are not handled.
- As the network could be under full control of the intruder, it is assumed that all messages are broadcasted.
- The logic is not suited to deal with Zero Knowledge Proofs. (Criticism by Snekkenes [15]). Exponential key exchanges are also supported.

5. The Simulation and Modelling Features of the Tool

The simulation part of the tool is not necessary to obtain the verification proofs of a protocol. It lets the protocol designer assess the functionality of a protocol based on the seeing and belief mechanisms of the logic. Three concepts can be simulated with the tool:

Finite State Machines FSMs:
Principals/Insiders and any kind of servers (application as well as ps/kdc) even with multiple processes running in parallel can be defined. These state machine definitions may not only contain the transitions for successful protocol runs, but also the different host's behaviour towards *meddling attacks* (≡ equivalent to physical transmission errors), *exhaustion* (≡ time–outs), *starvation* etc.

With this FSM concept, the characteristic of Kerberos described by Bellovin and Merritt [16] can be simulated whereby an intruder is capable of accumulating the equivalent of /etc/passwd even without eavesdropping by using the authentication server as an oracle on all login names.

Active intruders
The attacks described above as well as replays, transmission of old data or messages with outdated keys can be modelled with the tool. Several kinds of attacks can be mapped onto each other and are therefore equivalent. A methodology to have the tool automatically exploring promising attacks in the case of a verification proof failure and illustrating eventual functional flaws to the designer hasn't been defined yet, but some results of the Interrogator approach [8] may be helpful to do that.

Time

The concept of logical clocks [17] is available in the tool and eases realistic modelling of time–outs etc.

Recall that the logic takes into account several notions of time in parallel:

- with respect to the freshness attributes and to the respective logical postulates, a session is one single clock–tick. Principals remember nothing from previous sessions.
- within a session, the time and, thus, the beliefs of a principal and visibilities monotonically increase. This is best achieved with the concept of logical clocks.

Snekkenes [15] criticized that the logic is susceptible to step permutation. The new message format encompasses the logical clock concept. Therefore, temporally consistent and monotonic increase of beliefs and visibility is guaranteed.

6. The Implementation in Prolog and its Usage

Prolog answers queries with constructive proofs through resolution. These constructive proofs are employed in two different ways:

- A query can be given to Prolog and if it can be derived from the rules and facts already known to the system, it will answer with yes or no. For example: "Does a principal believes in the shared session key with another principal after the protocol run?" Successful resolutions of postulates are displayed which provides a detailed verification proof step–by–step.
- The query presented to the system can also contain variables. The system can now be programmed to search *all* valid instantiations of these variables. Thus, queries with variables are questions to the system which give all answers according to already known facts and clauses. With this user interface the tool benefits from full power of the Prolog. An interactive and stepwise development of protocols is now made possible since the system can be ordered to list all the beliefs and visible possessions of any principal at any point in time etc. Therefore, missing or redundant parts as well as faults of a protocol can be detected easily.

When modelling the protocol with FSMs every message emitting state transition is displayed. At the end or at any time during the protocol run the designer can use these messages to obtain the above mentioned listings of beliefs or verification proofs. Such beliefs or visibilities can also be included as preconditions for transitions of the FSMs even when these items are not necessary for generating the following message.

The new tool has been tested on many known protocols and all the BAN examples and has reported correct results.

7 Summary, Conclusions and Future Work

This research started from the observation that present tools to analyze the security of authentication protocols do not use known authentication logics. Moreover, they are sometimes incomplete and difficult to use.

An authentication logic was implemented in Prolog avoiding some shortcomings of the original BAN–logic. This required an in–depth analysis of BAN and its derivatives and revealed some new problems as, for example, the freshness re–definition in freshcreate and freshuse.

The inclusion of a finite state machine modelling on the functional side of authentication protocols enriches the new logic for the design of secure authentication protocols.

Directions for Future Work.
The tool could be extended to include the analysis of verifiable plaintext in combination with poorly chosen keys. Furthermore, the qualities of good keys could be made explicit, and new encryption schemes like Zero Knowledge Proofs, one way functions, exponential key exchange etc. could be added.

A revision of the logical postulates and constructs is desirable to deal more analytically with non–disclosure and (cooperating) insiders etc. and the directedness of authentication. On the functionality side, the detection and reporting of the way of active intruding actions should be automated.

References

1. Michael Burrows, Martin Abadi, and Roger Needham, "A Logic of Authentication", Report #39, Feb. 1989, Report of Systems Research Center, DEC Palo Alto, California; also in Operating Systems Review, Dec 3–6, 1989, Vol. 23, # 5, pp 1–13.
2. Li Gong, Roger Needham, and Raphael Yahalom, "Reasoning about Belief in Cryptographic Protocols", Proceedings of the 1991 IEEE Symposium on Research in Security and Privacy, May 20–22, 1991, Oakland California, pp. 234–248.
3. R. M. Needham and M. D. Schroeder, "Using Encryption for Authentication in Large Networks of Computers", CACM vol. 21, no 12, Dec 1978, pp. 993–999.
4. D. E. Denning, G. M. Sacco, "Timestamps in Key Distribution Protocols", CACM 24, 1981 pp. 533.
5 R. K. Bauer, T. A. Berson, and R. J. Feiertag, "A Key Distribution Protocol using Event Markers", ACM Transactions on Computer Systems, vol. 1, no 3, Aug. 1983, pp. 249–255.
6. T. Mark A. Lomas, Li Gong, Jerome H. Saltzer, Roger M. Needham, "Reducing Risks from Poorly Chosen Keys", Proceedings of the 12th ACM Symposium on Operating Systems Principles, Dec 1989, pp 14–18.
7. Richard A. Kemmerer, "Analyzing Encryption Protocols Using Formal Verification Techniques", IEEE Journal on Selected Areas in Communications, Vol. 7,4, May 1989, p 448–457.
8. J. K. Millen, S. C. Clark, and S. B. Freedman, "The Interrogator: Protocol Security Analysis", IEEE Transactions on Software Engineering, vol. 13, no 2, Feb. 1987, pp. 274–288.

9. R. E. Soper, E. S. Lee, P. I. P. Boulton, M. Stumm, B. Thomson, "Protocol Verification in a Trusted Network Architecture", Technical Report CSRI–236, October 1989.
10. Jennifer G. Steiner, Clifford Neuman, Jeffrey I. Schiller, "Kerberos: An Authentication Service for Open Network Systems", Proceedings of the USENIX Winter Conference, Feb. 1988.
11. D. Nesset, "A Critique of the Burrows, Abadi, and Needham Logic", Operating Systems Review, vol. 24, no. 2, April 1990, pp. 35–38.
12. E. S. Lee, P. I. P. Boulton, D. M. Lewis, M. Stumm, and B. Thompson, "A Trusted Network Architecture", Technical Report, Computer Systems Research Institute, University of Toronto, Oct. 1988, pp. 97.
13. National Bureau of Standards, Federal Information Processing Standards, National Bureau of Standards, Publication 46, 1977.
14. R. L. Rivest, A. Shamir, and L Adleman, "A Method for Obtaining Digital Signatures and Public–key Cryptosystems", Communications of the ACM Vol. 21, No. 2 Feb. 1978, pp. 120–126
15. E. Snekkenes, "Exploring the BAN Approach to Protocol Analysis" Proceedings of the 1991 IEEE CS Symposium on Research in Security and Privacy, May 20–22, 1991, Oakland California, pp. 171–181.
16. Steven M. Bellovin, Michael Merritt, "Limitations of the Kerberos Authentication System", Proceedings of the USENIX Winter 1991 Conference, Dallas, TX, earlier version in Computer Communications Review, vol. 20, no 5,October 1990, pp. 119–132.
17. Leslie Lamport, "Time, clocks and the ordering of events in a distributed system", CACM, 21(7), July 1978, pp. 558–565.