# Cryptographic Protocols over Open Distributed Systems: A Taxonomy of Flaws and related Protocol Analysis Tools[*][+]

**S.Gritzalis [1,2], D.Spinellis [3,4]**

[1] Department of Informatics, University of Athens,

TYPA Buildings, Athens GR-15771, GREECE

[2] Department of Informatics

Technological Educational Institute of Athens (T.E.I.) ofAthens,

Ag.Spiridonos St. Aegaleo GR-12210, GREECE

email: *sgritz@teia.ariadne-t.gr*

[3] Department of Mathematics, University of the Aegean,

Samos, GR-83200, GREECE

email: *dspin@aegean.gr*

[4] SENA SA

Byzantiou 2 N.Ionia GR-14234, GREECE

email: *dds@senanet.com*

**Abstract**

When designing and implementing cryptographic protocols one must avoid a number of possible flaws. In this paper we divide possible flaws based on the flaw pathology and the corresponding attack method, into elementary protocol flaws, password/key guessing flaws, stale message flaws, parallel session flaws, internal protocol flaws, and cryptosystem flaws. We then outline and comment on different attack construction and inference-based formal methods, protocol analysis tools, and process integration techniques and their effectiveness in aiding the cryptographic protocol design process by discovering protocol flaws with regard to the aforementioned proposed taxonomy of them.

[*] In Peter Daniel, editor, *16th International Conference on Computer Safety, Reliability and Security: SAFECOMP '97*, pages 123-137, York, UK, September 1997. European Workshop on Industrial Computer Systems: TC-7, Springer Verlag.

[+] This is a machine-readable rendering of a working paper draft that led to a publication. The publication should always be cited in preference to this draft using the reference in the previous footnote. This material is presented to ensure timely dissemination of scholarly and technical work. Copyright and all rights therein are retained by authors or by other copyright holders. All persons copying this information are expected to adhere to the terms and constraints invoked by each author's copyright. In most cases, these works may not be reposted without the explicit permission of the copyright holder

# 1 Introduction

A protocol is a set of rules and conventions that define the communication framework between two or more parties. The parties are said to be communicating *(principals)* and can be end-users, processes or computing systems. In cryptographic protocols part of at least one message is encrypted.

When developing a cryptographic protocol it is desirable to uncover any flaws as soon as possible. These flaws can occur because of incomplete or erroneous specifications. However, even correct specifications do not necessarily guarantee the correctness of a given implementation. Generally we can distinguish between three categories of cryptographic protocol flaws [1] according to the flaw source:

- *functional specification flaws* occur due to a logical flaw in the protocol's high level specification,
- *implementation-dependent flaws* [2] appear when a protocol's specification can result in implementations of which at least one exhibits the flaw and at least one other does not, and
- *implementation flaws*, are those faults that occur when a correct specification is incorrectly implemented.

# 2 A Taxonomy of Cryptographic Protocol Faults

After a thorough study of the flaws belonging to the aforementioned general categories we propose the following more detailed taxonomy of these flaws based on the flaw pathology and the corresponding attack method:

[1] Elementary protocol flaws

[2] Password/key guessing flaws

[3] Stale message flaws

[4] Parallel session flaws

[5] Internal protocol flaws

[6] Cryptosystem flaws.

## 2.1 Elementary protocol flaws

In the elementary flaw category belong all flaws that occur in protocols providing minimal or no protection against adversary attacks.

The flaw of the protocol proposed by [3] for authentication key exchange between two communication parties belongs to this category [1]. The session key is signed by A's private key before being sent to B. The flaw in this case is that a signature is used to provide message confidentiality. Similar problems [1] [4] appear in the CCITT X.509 authentication protocol [5]. The cause behind the most important of them is that the messages are encrypted before being signed making it therefore possible for an adversary to masquerade as the sender by changing the initial signature with his own.

## 2.2 Password/key guessing flaws

The flaws belonging to this category occur because users often choose their passwords from a small set of common words [6] [7]. In addition, in cases where a protocol uses a pseudo-random key, it is possible that the key is constructed in a way that can be reproduced by an adversary. As a result in case of an exhaustive key search attack the adversary can use a restricted probable password key space instead of the - much larger - possible key space. For this reason attacks based on this flaw category are referred to [8] as *dictionary attacks* or as *verifiable-text attacks*.

The user-supplied passwords could be rejected if they occur in a dictionary or consist of too few characters. The smallest allowable password size $P_{size}$ can be calculated [9] depending on the password alphabet size $A_{size}$, the required password life time $P_{life}$, the maximum rate at which passwords can be tried $G_{rate}$ and the maximum password guessing probability $G_{prob}$ and is given by the relationship:

$$P_{size} = log\ (P_{life}\ x\ G_{rate}\ /\ G_{prob})\ /\ log\ A_{size}$$

Password guessing attacks can be divided into three categories:

- *Detectable on-line password guessing attacks*: Every unsuccessful attempt is detected and logged by the authentication server S. After a specific number of unsuccessful attempts S will stop servicing the attacked password (thereby creating a denial of service vulnerability).
- *Undetectable on-line password guessing attacks*: In this attack mode [10], the attacker is trying to use a password that could be correct for an on-line transaction. The attacker gradually verifies the password’s correctness from the responses elicited from S. If the guess is incorrect, then the transaction is aborted; the next guess will be tried in a new transaction. A failed attempt can not be detected and logged by S, because S can not distinguish between a genuine and a password guessing transaction.

- *Off-line password guessing attacks*: The attacker is using authentication protocol message copies, guessing the password and verifying it in an off-line environment. S is not participating and therefore the procedure can not be detected.

Authentication protocols can be strengthened by introducing two basic requirements:

- the authentication server is to respond only to fresh requests and
- the authentication server is to respond only to requests of verifiable authenticity.

These requirements are vital for dealing with detectable on-line password guessing attacks [11], but are not relevant in relation to off-line attacks [12].

A number of protocols have been proposed for dealing with on-line [13] and off-line [14] [15] [12] password guessing attacks. In addition, two tools have been proposed for helping users pick stronger passwords [16]: *password generators* and *password monitoring programs*. Password generators are programs that are made available on systems in an effort to ensure "good password choices", which means that the selected password is difficult to guess, and easy for the user to remember. These programs have to be sufficiently random in the specific method in which they select password. Password monitors are programs that accept a user's choice for a password based on how likely it is that the password could be guessed. More sophisticated password monitors may ensure the password is not a known easy-to-guess one, check that this is not in a dictionary, analyse it to see if it looks too much like a real word, and use an addition process named "password guesser", in order to try and guess this password.

## 2.3 Stale message flaws

Often an adversary instead of a direct attack on a security protocol will try to utilise genuine protocol message fragments that he can neither read nor legally create. For this reason a lot of effort has been put into designing protocols that are not vulnerable to replay attacks.

Studying message replay attacks [17] has proposed a taxonomy based on the *message origin* and the *message destination*.

### *2.3.1. Message origin attacks*

In r*un external attacks* message fragments from one protocol run are used in another run. An example of such an attack [18] is based on the secret key Needham-Schroeder [19] protocol where the attacker can read the third protocol message:

[1] A $\rightarrow$ S : $A, B, N_a$

[2] S $\rightarrow$ A : $\{N_a, B, K_{ab}, \{K_{ab}, A\}_{Kb}\}_{Ka}$

[3] A $\rightarrow$ B : $\{ K_{ab}, A\}_{Kb}$

[4] B $\rightarrow$ A : $\{N_b\}_{Kab}$

[5] A $\rightarrow$ B : $\{N_b\text{-}1\}_{Kab}$

and having enough time and processing power can guess the session key and use it in a next protocol run as the original communicating parties will not know that the session key has been compromised. This attack is of course only viable when there are no mechanisms for outdating session keys. This attack is a run external attack because during a protocol run a message from a previous run was used. A parallel protocol run was not needed.

A parallel protocol run can also lead to a successful attack [17]. The BAN-Yahalom [4] protocol contains the following steps:

[1] A $\rightarrow$ B :$A, N_a$

[2] B $\rightarrow$ S : $B, \{A, N_a\}_{Kbs}, N_b$

[3] S $\rightarrow$ A :$\{B, N_a, K_{ab}\}_{Kas}, \{A, K_{ab}, N_b\}_{Kbs}, N_b$

[4] A $\rightarrow$ B :$\{A, K_{ab}, N_b\}_{Kbs}, \{N_b\}_{Kab}$

When *Eve* is performing an attack masquerading as *A-Alice*, after the protocol's step 2 she starts a parallel protocol run:

[1] A $\rightarrow$ B :$A, N_a$

[2] B $\rightarrow$ S : $B, \{A, N_a\}_{Kbs}, N_b$

[1'] $E_a \rightarrow$ B: $A, (N_a, N_b)$

[2'] B $\rightarrow E_s$: $B, \{A, N_a, N_b\}_{Kbs}, N'_b$

[3] ............

[4] $E_a \rightarrow$ B : $\{A, N_a\ (=K_{ab}), N_b\}_{Kbs}, \{N_b\}_{Kab}$

*Eve* is using in the second run the concatenation of $N_a$ and $N_b$ as a *nonce*. As soon as *Eve* receives the encrypted message from step 2 of the second run she is using it as the first encrypted part of step 4 of the first run. In the end *Eve* has masqueraded as *A-Alice* to *B-Bob* and received the session key. This attack is also a run external attack because during a protocol run a message from a previous run is used. In this case however, the attack was based on a parallel protocol run.

*Run internal attacks* are using message fragments from the same protocol run. Such an attack [20] on the Neuman-Stubblebine protocol [21] contains the following steps:

[1] A $\rightarrow$ B : $A, N_a$

[2] B $\rightarrow$ S : $B, \{A, N_a, T_b\}_{Kbs}, N_b$

[3] S $\rightarrow$ A : $\{B, N_a, K_{ab}, T_b,\}_{Kas}, \{A, K_{ab}, T_b\}_{Kbs}, N_b$

[4] A $\rightarrow$ B : $\{A, K_{ab}, T_b\}_{Kbs}, \{N_b\}_{Kab}$

During the protocol run, the attacker *Eve*, masquerading as *A-Alice*, receives a part of message 2 and is using it to construct message 4.

[4'] $E_a$ $\rightarrow$ B : $\{A, N_a\ (=K_{ab}), T_b\}_{Kbs}, \{N_b\}_{Na(=Kab)}$

The new message 4 is the same as message 2, but the session key has been changed with the nonce $N_a$. In this way the last part of the protocol's step 4 was correctly implemented and therefore *Eve* could run a session with *B-Bob* masquerading as *A-Alice*, and make *B-Bob* accept the session key that belongs to *Eve*. This attack is a run internal attack because during a protocol run a message from the same run is used.

### *2.3.2 Message destination attacks*

One other attack [20] on the previously discussed BAN-Yahalom protocol is the following:

[1] A $\rightarrow$ $E_b$: $A, N_a$

[1'] $E_b$ $\rightarrow$ A: $B, N_a$

[2'] A $\rightarrow$ $E_s$: $A, \{B, N_a\}_{Kas}, N'_a$

[2''] $E_a$ $\rightarrow$ S: $A, \{B, N_a\}_{Kas}, N_a$

[3'] S $\rightarrow$ $E_b$: $\{A, N_a, K_{ab}\}_{Kbs}, \{B, K_{ab}, N_a\}_{Kas}, N_a$

[2] ............

[3] $E_s$ $\rightarrow$ A: $N_i, \{B, K_{ab}, N_a\}K_{as}, \{A, K_{ab}, N_a\}K_{bs}$

[4] $E_a$ $\rightarrow$ B: $\{A, K_{ab}, N_a\}_{Kbs}, \{N_i\}_{Kab}$

This attack demonstrates a *message reflection* problem, i.e. the return of a message to the original sender. *Straight replays* of message 2' to message 2'', are those where the message is sent from the sender to the supposed receiver even though the message semantics are not preserved (text has been added or the message has been delayed). *Message deflection* of message 3' to message 3, occurs when protocol exchange messages are redirected to a third entity.

## 2.4 Parallel session flaws

*Parallel session attacks (*or *oracle session attacks, multi-role flaws*) are flaws that allow an adversary to gain the desired information by exchanging suitable protocol messages.

Participants in these protocols can be distinguished [22] either as *single role*, or as *multi-role* participants. In single role protocols there is a *one to one* relationship between a participant and his role. In a multi role protocol this relationship is a *one to many*. In both cases a participant's presence can only be interpreted as a specific *role* and not as the specific participant's *name.* Therefore a participant *p* can at different times act in role *A* and role *B*. It can be proven [22] that any analysis method that fails to distinguish between the possible *roles* of a participant and the participant's *name* will not yield dependable results.

In the following paragraphs we will study a parallel session single role flaw and a parallel session multi role flaw [1] using the *three-pass protocol* [23]:

[1] A $\rightarrow$ B : $\{M\}_{Ka}$

[2] B $\rightarrow$ A : $\{\{M\}_{Ka}\}_{Kb}$

[3] A $\rightarrow$ B : $\{M\}_{Kb}$

The protocol can be used for transferring a secret message without the use of a trusted third party. It does however not provide authentication as *A* and *B* do not share any secrets.

The protocol utilises a cryptographic *commutative function*, which satisfies the relationship:

$$\{\{M\}_{Ka}\}_{Kb} = \{\{M\}_{Kb}\}_{Ka}.$$

### *2.4.1 Parallel session single role flaws*

In a single role run of the protocol the following situation can occur [1]:

[1] Alice $\rightarrow$ Eve $_{Bob}$ : $\{M\}_{Ka}$

[2] Eve $_{\text{Bob}}$ $\rightarrow$ Alice : $\{M\}_{Ka}$

[3] Alice $\rightarrow$ Eve $_{\text{Bob}}$ : $M$

Participant *A-Alice*, sends a request to *B-Bob*. The message is however intercepted by *Eve* who masquerades as *Bob* and uses steps 2 and 3 for intercepting the cleartext of the secret message *M*. This attack could have been prevented if participant *A* had a way to distinguish between different message types and could therefore prevent the transmission of messages of type *unencrypted*.

### *2.4.2 Parallel session multi role flaws*

The protocol can be used by multi role participants as follows:

[1.1] Alice $_{\text{A}}$ $\rightarrow$ Eve $_{\text{Bob B}}$ : $\{M\}_{Ka}$

[2.1] Eve $_{\text{Bob A}}$ $\rightarrow$ Alice $_{\text{B}}$ : $\{M\}_{Ka}$

[2.2] Alice $_{\text{B}}$ $\rightarrow$ Eve $_{\text{Bob A}}$ : $M$

[1.2] Eve $_{\text{Bob B}}$ $\rightarrow$ Alice $_{\text{A}}$ : *any text*

[1.3] Alice $_{\text{A}}$ $\rightarrow$ Eve $_{\text{Bob B}}$ : $\{any\ text\}_{Ka}$

In this case [1], after step 1.1 *Eve* intercepts the message from *Alice*. After step 2.1 *Eve* establishes a new session masquerading as $Bob_A$ in order to return to $Alice_B$ the message that was sent by $Alice_A$. In step 2.2 $Alice_B$ returns the message *M* decrypted which is of course received by *Eve*. *Eve's* mission has been accomplished since she is now in possession of a decrypted version of *M*. *Eve* can potentially complete the session so that *Alice* will not realise that the message has been intercepted.

## 2.5 Internal protocol flaws

*Internal protocol flaws* occur when at least one of the protocol participants fails to complete all requisite actions.

A typical example of this flaw [1] is step 3 of the three pass protocol. Before the message is sent it is desirable for the participant *A* to ensure that the message is encrypted. As mentioned above, this requirement should be part of the protocol specifications and implementors should always ensure that it is always satisfied.

## 2.6 Cryptosystem flaws

Encryption algorithms and related protocols are designed and used in order to satisfy some data confidentiality or authentication requirements. A specific implementation may satisfy all properties required by the algorithm and the protocol specification, but exhibit additional properties that compromise the confidentiality or authentication requirements. In that case *cryptosystem - related flaws* [24] [25] [1] are said to occur.

Often a poor implementation of a given cryptosystem is all that is needed in order to compromise it. [25] details a number of protocols based either on public key algorithms (e.g. the *low entropy protocol*) or on secret key algorithms (e.g. the *single key protocol*) that exhibit such flaws.

# 3 FORMAL CRYPTOGRAPHIC PROTOCOL ANALYSIS AND DOCUMENTATION METHODS

## 3.1 Introduction

In the last decade a number of methods and tools have been published and implemented that detect cryptographic protocol flaws by analysing and documenting their operation [26]. The most important methods can be divided into two categories [27] according to their operation domain:

*Attack-construction tools* construct probable attack sets based on the protocol's algorithms algebraic properties. These methods [28] [29] [30] [31] [32] [33] are targeted towards ensuring authentication, correctness or security properties and are not dependent on the correctness of a proposed logic. Their disadvantage lies mainly in the big number of possible events that must be examined.

*Inference-construction tools* are utilising either *modal logic*, *logic of knowledge*, or *logic of belief*. These methods [4] [34] [35] include belief logics which are potentially much faster, capable of analysing large, complicated protocols that the attack-construction tools are incapable of analysing in a reasonable time, and are widely used. A number of specific problems associated with them [27] [36] [37] [20] [38] range from their inability to analyse zero knowledge protocols or to address only authentication  or to detect parallel session multi-role flaws to the difficulty of transforming messages and prepositions to idealised messages.

## 3.2 Flaw detection by attack construction tools

Flaw construction tools can be distinguished into three categories based on their theoretical foundation. These categories are:

### *3.2.1 Methods based on validation languages and tools that are not specifically developed for analysing cryptographic protocols.*

These methods analyse a cryptographic protocol as any other program whose correctness they are trying to prove. This is done by specifying the protocol: as a finite-state machine [32] [33], using predicate calculus [29], or within a process algebra [39] [40].

Some researchers [32] [33] map the protocol to a finite-state machine. The analysis method proposed by [32], verifies the basic properties of a number of protocols, detects basic flaws, but can not detect flaws due to the re-use of old messages as no temporal assumptions are used. The method proposed by [33] also verifies the basic properties of a number of protocols, but exhibits a number of problems as the number of states increases. In addition, in order to deal with flaws related to the re-use of old messages the author proposes to incorporate into the analysis data from the session key message contents.

Another approach [29] is based on predicate calculus extensions. This method is using the specification language Ina Jo [41] and the Formal Development Methodology (FDM). Formal specifications written in Ina Jo specify definitions, initial conditions, transforms, axioms, and criteria. Criteria are used to specify critical requirements for a secure state. Ina Jo formal specifications can then be executed and verified by tools such as Inatest. This approach has been successful in locating both active and passive attack flaws, since in both cases the intruder is a separate entity in the model's mathematical framework.

A more recent approach [39] [40] is based on modelling the communicating principals and the intruder as CSP processes. The proposed method can be used to formalise messages, traces, intruders, and nonce challenges. The Failures Divergence's Refinement Checker (FDR) tool is a general purpose tool that can be used to determine whether an implementation refines a specification. In the case of protocol authentication, checking for refinement amounts to testing whether each trace of the implementation is also a trace of the specification.

Although these methods have been judged as an important contribution to the field, research has turned into more specialised directions. The driving force behind this turn is the desire to use cryptography domain specific reasoning knowledge.

### *3.2.2 Expert system, scenario based methods*

The method due to [31], known as the Interrogator Model, is using a system based on a Prolog solver to guide the designer towards examining whether a specific protocol can lead to an undesirable situation, such as compromising a key. Although this method can not guarantee absolute safety, it works very well in identifying specific protocol flaws.

The method has been successfully used to find various known flaws in protocols such as the [42] [19] [43] [44] and [45]. No previously undetected vulnerabilities in well known protocols have been discovered using this method. The tool's applicability is limited by the operators it supports (conventional and public key encryption, exclusive-or and limited finite-field exponentiation).

#### *3.2.3. Algebraic simplification theoretic model methods*

Important methods in this category have been proposed by [28] [46] and [30]. Among them the NRL Protocol Analyser [30] is believed to be the most promising method of assuring correctness in cryptographic protocols. This method specifies the protocol and its analysis as a set of transition rules governing the actions of honest principals as well as rules describing possible - non intruder caused - system failures, a set of operations available to the principals, and rewrite rules obeyed by the operations.

The NRL Protocol Analyser has been successfully used to uncover known flaws of all our proposed taxonomy types, especially stale message flaws. The NRL Protocol Analyser has also been used to locate a series of previously unknown flaws in a number of protocols [45]. The current implementation's main drawback is the paucity of reduction operators which are limited to conventional and public key encryption operators. In addition, as with most rule rewrite systems, it is not clear how well the system scales as more complicated algorithms will need to be expressed using an ever increasing set of rules.

## 3.3 Inference based methods

Inference based methods are based on formal protocol specification modelling using the Logics of Knowledge and Trust. A representative such method, BAN Logic [4], is widely used for authentication protocol verification. BAN Logic considers authentication as a function of message freshness and integrity and is using a formal model for the authentication protocol messages based a predefined set of axioms.

BAN Logic has been successfully used to uncover a number of unknown flaws [5] [19] [47] as well as superfluous operations in widely used protocols [5] [48] [19] [49] [47]. BAN Logic can not be extended to zero knowledge protocols [20], and can not detect parallel session multiple-role flaws nor stale reflected message flaws [37], although it can detect run external attack flaws [4]. Furthermore, BAN Logic does not cover implementation-related flaws such as those included in [21] and detected in [2].

A number of other alternative logics have been proposed correcting or extending the existing framework [34] [35]. GNY Logic includes a parser that can detect whether a message has been sent in the past. However, even this extension does not completely detect stale message flaws.

The most important drawback of BAN-type logics is the lack of strict application techniques for converting messages and beliefs into idealised messages. A number of improvements have been suggested [50] [36] to deal with this problem. Despite this problem, inference based methods, and BAN Logic in particular is used in many new protocol specifications as for example in the analysis, specification and verification of Internet commercial transaction protocols [51] [52]. As BAN Logic can not prove that a protocol is secure, but can provide information about the possible occurrence of undesirable properties it can be used as a complement to the NRL Protocol Analyser [30].

### 3.4 Design process integration aspects

The multitude of protocol analysis approaches, methods, and tools hinders their integration into the protocol design process. Every different protocol analysis tool provides its own formal specification language; different from the message-oriented protocol descriptions that are typically published. Two approaches have been proposed in order to bridge the gap between the protocol analysis formalisms and the protocol design process.

One approach [53], proposes the use of an Interface Specification Language (ISL) in order to allow arbitrary protocol design processes to interface to the analysis tool. This approach has been used to provide a front end to the Automatic Authentication Protocol Analyser (AAPA) [54] [55] a tool that uses an extension of the GNY logic for proving protocol properties.

A second approach [56], proposes the use of a Common Protocol Specification Language (CAPSL) to bridge the gap between the typical informal presentations of protocols given in papers and the precise characterisations required to conduct formal analysis. The proposers of this approach hope that proponents of different analysis techniques will offer algorithms for compiling the CAPSL language into whatever form they require making it therefore possible to directly compare technique protocol assumptions and analysis results. This work is in progress, has not yet been completed, and it is described in a WWW site for suggestions, refinement and standardisation of the language definition.

## 4 Conclusions

Having examined a number of cryptographic protocol flaws we provided a possible taxonomy based on the flaw pathology and the corresponding attack method: exploitation of protocol or implementation weaknesses, password/key guessing, message re-use, or the establishment of a parallel session. The use of formal methods can definitely aid in the analysis, verification, validation, and security valuation of existing and proposed cryptographic protocols. As distributed systems and open interconnected networks are increasingly being used for transactions of commercial value, the transfer of sensitive personal data, and as society's infrastructure fabric increasingly depends on them the formal analysis of cryptographic protocols will be an important research topic.

The outlined presentation of general purpose formal analysis tools used in the cryptographic protocol domain as well as domain specific approaches presented in section 3 is an initial attempt at categorising tools and providing our view of their relative strengths and weaknesses with regard to the aforementioned proposed taxonomy of cryptographic protocols flaws. We believe that in the coming years formal method based tools will increasingly be used during cryptographic protocol design process, especially in the initial stages of the whole process.

**Acknowledgements**

The authors would like to thank Prof. P.Georgiadis, University of Athens, Greece, for his helpful comments.

## 5 References

1. Carlsen U. Cryptographic Protocol Flaws. In: *Proceedings of the 1994 IEEE Computer Security Foundations Workshop VII*. IEEE Computer Society Press, 1994, pp. 192-200
2. Carlsen U. Using Logics to Detect Implementation-Dependent Flaws. In: *Proceedings of the 9th IEEE Annual Computer Security Applications Conference*. IEEE Computer Society Press, 1993, pp. 64-73
3. Nesset D. A Critique of the BAN Logic. *ACM Operating Systems Review* 1990; 24(2) 35-38
4. Burrows M., Abadi M., Needham R. A Logic of Authentication. *ACM Transactions on Computer Systems* 1990; 8(1) 18-36
5. CCITT. *CCITT X.509: The Directory - An Authentication framework*. CCITT, 1988
6. Morris R. Password Security: A Case History. *Communications of the ACM* 1979; 22(11) 594-597
7. Klein D. Foiling the Cracker: A Survey of, and Improvements to, Password Security. In: *Proceedings of the USENIX Security Workshop II*. USENIX Association, 1990, pp. 5-14
8. Gong L. Attacks in Cryptographic Protocols. In: *Proceedings of IEEE INFOCOM '90*. IEEE Computer Security Society Press, 1990
9. Janson P., Molva R. Security in Open Networks and Distributed Systems, *Computer Networks and ISDN Systems 1991;* 22(5) 323-346
10. Ding Y., Horster P. Undetectable on-line password guessing attacks, *ACM Operating Systems Review* 1995; Vol. 29, No. 4, 77-86
11. Tsudik G., Van Herreweghen E. Some Remarks on Protecting Weak Keys and Poorly-Chosen Secrets from Guessing Attacks. In: *Proceedings of the 12th IEEE Symposium on*

*Reliable Distributed Systems.* IEEE Computer Society Press, 1993, pp. 136-141

12. Gong L. Optimal Authentication Protocols Resistant to Password Guessing Attacks. In: *Proceedings of the 1995 IEEE Computer Security Foundations Workshop VIII*. IEEE Computer Society Press, 1995, pp. 24-29
13. Tardo J., Alagappan K. SPX: Global Authentication Using Public Key Certificates. In: *Proceedings of the 1991 IEEE Symposium on Research in Security and Privacy.* IEEE Computer Society Press, 1991, pp. 23-244
14. Bellovin S., Merritt M. Encrypted Key Exchange: Password-Based Protocols Secure against Dictionary Attacks. In: *Proceedings of the 1992 IEEE Symposium on Security and Privacy.* IEEE Computer Society Press, 1992, pp. 72-84
15. Gong L., Lomas M., Needham R. Saltzer J. Protecting Poorly Chosen Secrets from Guessing Attacks. *IEEE Journal on Selected Areas in Communications 1993;* Vol. 11, No. 5, 648-656
16. Jobusch D., Oldehoeft A. A survey of Password Mechanisms: Weaknesses and Potential Improvements. *Computers and Security 1989;* Vol. 8, No. 7, 587-603
17. Syverson P. A Taxonomy of Replay Attacks. In: *Proceedings of the 1994 IEEE Computer Security Foundations Workshop VII*. IEEE Computer Society Press, 1994, pp. 187-191
18. Denning D., Sacco G. Timestamps in Key Distribution Protocols. *Communications of the ACM 1981;* Vol. 24, No. 8, 533-536
19. Needham R., Schroeder M. Using Encryption for Authentication in large networks of computers. *Communications of the ACM 1978;* 21(12) 993-999
20. Syverson P. On Key Distribution Protocols for Repeated Authentication. *ACM Operating Systems Review 1993;* 27(4) 24-30
21. Neuman B., Stubblebine S. A Note on the Use of Timestamps as Nonces. *ACM Operating Systems Review 1993;* 27(2) 10-14
22. Snekkenes E. Roles in Cryptographic Protocols. In: *Proceedings of the 1992 IEEE Computer Security Symposium on Security and Privacy*. IEEE Computer Society Press, 1992, pp. 105-120
23. Shamir A., Rivest R., Adleman L. Mental Poker. *MIT Laboratory for Computer Science*, 1978, Report TM-125: 178-184
24. Massey J. An Introduction to Contemporary Cryptology. In: *Proceedings of the IEEE*. IEEE Computer Society Press, 1988, Vol. 76, No. 5, pp. 533-549
25. Moore J. Protocol Failures in Cryptosystems. In: *Proceedings of the IEEE*. IEEE Computer Society Press, 1988, Vol. 76, No. 5, pp. 594-602
26. Kemmerer R., Meadows C., Millen J. Three Systems for Cryptographic Protocol Analysis. *Journal of Cryprology* 1994; (7) 79-130
27. Brackin S. A. HOL Extension of GNY for Automatically Analysing Cryptographic Protocols. In: *Proceedings of the 1996 IEEE Computer Security Foundations Workshop IX*. IEEE Computer Society Press, 1996, pp. 62-76
28. Dolev D., Yao A. On the Security of Public Key Protocols. *IEEE Transactions on Information Theory 1983;* 29(2) 198-208
29. Kemmerer R. Analysing encryption protocols using formal verification techniques. *IEEE Journal on Selected Areas in Communications 1989;* 7(4) 448-457
30. Meadows C. Applying Formal Methods to the Analysis of a Key-Management Protocol.

*Journal of Computer Security 1992;* vol. 1, 5-35

31. Millen J. The Interrogator Model. In: *Proceedings of the 1995 IEEE Symposium on Security and Privacy.* IEEE Computer Society Press, 1995, pp. 251-260
32. Sidhu D. Authentication Protocols for Computer Networks. *Computer Networks and ISDN Systems 1986;* 11, 297-310
33. Varadharajan V. Verification of Network Security Protocols. *Computers and Security 1989;* Vol. 8, 693-708
34. Gong L., Needham R., Yahalom R. Reasoning about Belief in Cryptographic Protocols. In: *Proceedings of the 1990 IEEE Symposium on Security and Privacy.* IEEE Computer Society Press, 1990, pp. 234-248
35. Syverson P., van Oorschot P.C. On Unifying some Cryptographic Protocol Logics. In: *Proceedings of the 1994 IEEE Computer Security Foundations Workshop VII.* IEEE Computer Society Press, 1994, pp. 14-29
36. Gritzalis S. BAN logic for the analysis and verification of authentication protocols in distributed systems: A Review. In: *Proceedings of the 1st meeting of the IKAROS Human Network for the Safety, Quality, and Reliability in Information and Communication Technologies,* 1996, (in Greek)
37. Kessler V., Wedel G. AUTLOG-An advanced Logic of Authentication. In: *Proceedings of the 1994 IEEE Computer Security Foundations Workshop VII.* IEEE Computer Society Press, 1994, pp. 90-99
38. Syverson P. The Use of Logic in the Analysis of Cryptographic Protocols. In: *Proceedings of the 1991 IEEE Computer Security Symposium on Security and Privacy.* IEEE Computer Society Press, 1991, pp. 156-170
39. Roscoe, A. W. Modelling and verifying key-exchange protocols using CSP and FDR. In: *Proceedings of the 1995 IEEE Computer Security Foundations Workshop IIX.* IEEE Computer Society Press, 1995, pp. 98-107
40. Lowe D. Breaking and Fixing the Needham-Schroeder Public-Key Protocol Using FDR. In: *Proceedings of TACAS.* Springer Verlag, 1996, pp. 147-166
41. Scheid J., Holtsberg S. *Ina Jo Specification Language Reference Manual,* System Development Group, Unisys Corporation, CA, 1988
42. Diffie W., Hellman M. New Directions in Cryptography. *IEEE Transactions on Information Theory 1976;* Vol. IT-22, No. 6, 644-654
43. Tatebayashi M., Matsuzaki N., Newman D. Key Distribution Protocol for Digital Mobile Communications Systems. In: *Advances in Cryptology, CRYPTO '89.* Springer Verlag, 1989, pp. 324-333 (Lecture Notes in Computer Science no. 435)
44. Purdy G., Simmons G., Studier J. A Software Protection Scheme. In: *Proceedings of the 1982 IEEE Symposium on Security and Privacy.* IEEE Computer Society Press, pp. 99-103
45. Simmons G. How to Selectively Broadcast a Secret. In: *Proceedings of the 1985 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 1985
46. Syverson P. Knowledge, belief and Semantics in the Analysis of Cryptographic Protocols. *Journal of Computer Security 1992;* Vol. 1, No. 3 317-334
47. Satyanarayanan M. Integrating Security in a large distributed system. *ACM Transactions on Computer Systems 1989;* 7(3) 247-280

48. Millen J., Neuman C., Schiller J., Saltzer J. Kerberos Authentication and Authorisation system, *Project Athena Technical Plan*, Section E.2.1. M.I.T., 1987

49. Otway D., Rees O. Efficient and timely mutual authentication. *ACM Operating Systems Review 1987;*21(1) 8-10

50. Mao W. An Augmentation of BAN-like Logics. In: *Proceedings of the 1995 IEEE Computer Security Foundations Workshop VIII*. IEEE Computer Society Press, 1995, pp. 44-56

51. Bellare M., Garay J., Hauser R., et al. iKP - a family of secure electronic payment protocols. In: *Proceedings of the First USENIX Workshop on Electronic Commerce,* USENIX Association, 1995

52. Pal G. Verification of the iKP family of secure electronic payment protocols, *http://web.mit.edu/gnpal/www/ikp/verify_ikp.html,* 1996

53. Brackin S. An Interface Specification Language for Automatically Analysing Cryptographic Protocols. In: *Proceedings of the 1997 Symposium on Network and Distributed System Security.* IEEE Computer Society Press, 1997, pp. 40-51

54. Brackin S. Automatic Formal Analyses of Cryptographic Protocols. In: *Proceedings of the 19th National Conference on Information Systems Security,* IEEE Computer Society Press, 1996

55. Brackin S. Automatic Formal Analyses of Cryptographic Protocols, updated version of [54], private communication, 1997

56. Millen J. CAPSL - Common Authentication Protocol Specification Language, work in progress*: http://www.mitre.org/research/capsl/,* 1997