Analysing the Combined Kerberos Timed Authentication Protocol and Frequent Key Renewal Using CSP and Rank Functions

Yoney Kirsal-Ever[1], Agozie Eneh[2], Orhan Gemikonakli[1] and Leonardo Mostarda[3]
[1] Department of Computer Communications Engineering, School of Science and Technology,
Middlesex University, The Burroughs, Hendon, London, NW4 4BT, United Kingdom
[e-mail: y.kirsal,o.gemikonakli@mdx.ac.uk]
[2] Department of Computer Science, University of Nigeria, Nsukka, Nigeria
[e-mail: agozie.eneh@unn.edu.ng]
[3]School of Science and Technology, Camerino University,
Via Modanna delle Carceri, 9, 62032 Camerino, Italy
[email: leonardo.mostarda@unicam.it]
*Corresponding author: Yoney Kirsal-Ever

Received April 22, 2014; revised September 4, 2014; revised September 30, 2014; accepted October 2, 2014; published December 31, 2014

Abstract

Authentication mechanisms coupled with strong encryption techniques are used for network security purposes; however, given sufficient time, well-equipped intruders are successful for compromising system security. The authentication protocols often fail when they are analysed critically. Formal approaches have emerged to analyse protocol failures. In this study, Communicating Sequential Processes (CSP) which is an abstract language designed especially for the description of communication patterns is employed. Rank functions are also used for verification and analysis which are helpful to establish that some critical information is not available to the intruder. In order to establish this, by assigning a value or rank to each critical information, it is shown that all the critical information that can be generated within the network have a particular characterizing property. This paper presents an application of rank functions approach to an authentication protocol that combines delaying the decryption process with timed authentication while keys are dynamically renewed under pseudo-secure situations. The analysis and verification of authentication properties and results are presented and discussed.

Keywords: CASPER, CSP, Kerberos, Key-Exchange Protocol, Key Renewal, Network Security, Rank Functions, Timed Authentication Protocols

http://dx.doi.org/10.3837/tiis.2014.12.021

1. Introduction

Owning to the growing popularity and the use of computers and network-based electronic devices, providing privacy and data integrity has become more crucial; effective mechanisms are necessary to prevent network-based attacks and unauthorised access. For the purposes of attack prevention, authentication and access control play a vital role [1]. In order to meet increasing demands in secure computer communications, various security protocols have been developed.

Kerberos is one of these commonly used mechanisms. It is based on Needham-Schroeder Authentication Protocol [12]. It utilises symmetric key cryptography to provide authentication for client-server applications. The Kerberos architecture is divided into two core elements, Key Distribution Centre (KDC) and Ticket Granting Service (TGS). The KDC stores authentication information while TGS holds digital tickets for identifying clients and servers of networks.

A new protocol description is designed [7] as the initial step of developing a specific authentication protocol that provides authentication following a previously proposed framework [5]. This protocol has properties of Kerberos and Key-Exchange protocols together with a powerful intruder model. Although the intruder has been given power to attack, the protocol is successful in preventing replay attacks. The designed protocol adopts the challenge-response paradigm. The interactions between the entities are represented using numbers 1-19 [4]. These numbers represent the interactions between the legitimate entities of a wireless LAN environment. Interactions take place between the client and the access point, between the access point and the authentication server, between authentication server and KDC, authentication server and TGS, and between KDC and TGS.

A new approach has been proposed to shut down access to the authentication server for a period of time to enable the distribution of randomly generated keys to users in a relatively secure way [7]. Renewing keys at various intervals while potential intruders are blocked out would inevitably work against intruders. Although intruders may have the power to attack, the protocol is successful in preventing replay attacks [8].

In this research, a new protocol description is designed as the initial step of developing a specific authentication protocol, following a previously proposed framework [4]. Kerberos has been used as the platform in the development of this protocol which is based on the Key-Exchange protocol enhanced with a powerful intruder model. It is proven that the developed protocol description is sufficient despite the extensive attacking capabilities of the intruder. The components of the developed protocol description are validated and verified by rank function theorem similar to the approaches used in [6], [14], and [15]. This research presents an approach of dynamically renewing keys under pseudo-secure situations thus significantly reducing the chances of potential intruders. The proposed approach involves secure key distributions at various intervals. During key distribution, temporary interruption to link/server access is ensured. The access restriction happens for short intervals.

Please note that Kerberos with timed-delay, and delayed decryption properties are presented in [5] for continuously evolved threats of penetration and other forms of attacks. However the resulting protocol is still vulnerable to new forms of intruders. On the other hand in [6] dynamically renewing keys under pseudo-secure situations is introduced which provides secure key distributions at various intervals (During key distribution, temporary interruption to link/server access is ensured.), but protocol in [6] does not use timed-delay. Without the timed-delay, the intruders may break into the system before the key renewal takes place. The access restriction happens for short intervals. In this study the two approaches are combined in order to provide improved security. Schneider's CSP processes and rank function theorem [14] are applied to expose any flaws in the design. Possible attacks have been successfully identified. It has also been reported that, the protocols often fail when they are analysed critically [10]. At this point, formal methods emerged for verification of security protocols. One of the most preferred methods is the use of general purpose verification tools. The Communicating Sequential Processes (CSP) is one of these tools. It is an abstract language designed especially for the description of communication patterns of concurrent system components that interact through message passing. Schneider introduces the notion of rank functions to analyse the protocols by using the process algebra CSP.

Although it is proven that CSP is successful in finding attacks upon a number of protocols, description of a system needs time and substantial experience in order to avoid mistakes. In order to address these concerns, CASPER has been developed [11]. CASPER is a program that automatically produces a CSP description from a more abstract description, thus simplifying the modelling and analysis process. A CASPER script could be divided into two parts: a general part that specifies a model of a system running the protocol in question and a specific part that defines given functions; the parameters of the protocol.

Supportive ideas are also developing, to prove that formal methods are vital for validating cryptographic protocols and their relative simplicity, which makes them amenable to, automated analysis. Various results are provided in [2] to illustrate this. Addition to Cortier et.al [2], in their study Datta et. al. [17] proved security properties, such as secrecy of passwords and the keys, and non-injective agreements of network protocols that use public and symmetric key cryptography. This study is designed around a process calculus with actions for possible protocol steps including generating new random numbers, sending and receiving messages, and performing decryption and digital signature verification actions. Their study supports compositional reasoning about complex security protocols and has been applied to a number of industry standards including SSL/TLS, IEEE 802.11i and Kerberos V5.

In this work, introduced new protocol based on the use of timestamps to delaying decryption [8]. For protocol verification, Schneider's CSP process algebra is used along with the central rank function theorem. The proposed model is based on secure key distributions at various intervals. During key distribution, access to the network of servers is not allowed. The access restrictions happen for short intervals. The proposed protocol presents the initial steps for implementing a previously proposed framework [4]. In order to prevent intrusion, the proposed protocol stops user access for short intervals to ensure secure key distribution. It is assumed that attackers can sniff the exchange of keys and try to break the authentication. Because of this reason, re-exchange of the keys takes place in a secure line. Once the key distribution is completed the access is re-granted to the users. This attempt restricts the time for a potential attacker to find flaw which is sniffing the exchange of the keys, and the password, within the protocol.

2. Related Work

In order to provide higher security, various specifications regarding security protocols have been developed and many security protocols have been built around these specifications. Most of these protocols agreed upon a cryptographic key or achieved authentication specifications [1, 8].These studies indicate that cryptographic protocols are prone to various types of attacks.

In [2], an existing decision procedure for solving constraint systems is refined to prove their result. It is mentioned that several decision procedures already exist for solving constraint systems and additionally, some of them are based on a set of simplification rules allowing a general constraint system to be reduced to some simpler one.

Also, [2] shows how to safely compose secure protocols by tagging encryption, focusing on secrecy properties. It clearly states that whenever a protocol preserves the secrecy of some data, it still preserves secrecy when other tagged protocols are executed in the same environment. This property of formal methods is valid for this study as well.

Currently, formal methods, mathematically-based techniques for the specific, development and verification of software and hardware systems [22], are spreading through into every phase of protocol development. These phases are classified as design, specification and verification [22, 24].

Kerberos Authentication Protocol designed as part of project Athena, provides secret key cryptography [20]. It uses key distribution. Clients and servers use digital tickets to identify themselves to the network and secret cryptographic keys for secure communications.

Kerberos' operation is system and application independent. Kerberos provides a mutual authentication between a client and a server. It assumes that initial transactions take place on an open network where clients and servers may not be physically secure and packets travelling on the network can be monitored and even possibly modified. Kerberos is independent of the security features defined in IEEE 802.11.

In wireless networks, although Kerberos relies on the provisions of IEEE 802.1 x standards, owing to the fact that, its operation is system and application independent, security features for authentication are independent as well. Kerberos protocol assumes that initial transactions take place on an open network where clients and servers may not be physically secure and packets travelling on the network can be monitored and even possibly be modelled [18]. Zrelli and Shinoda designed the integration of Kerberos protocol as an authentication method in existing EAP-based authentication frameworks. They defined the architectural elements and their interactions, and the encapsulation of Kerberos messages in EAP packets are specified [19].

The framework’s three entities (the supplicant, the authenticator, and the authentication server) mutually authenticate each other prior to data traffic [4, 6]. It was built on the assumption that none of the parties should be trusted in a wireless local area network communication environment.

Kerberos's mutual authentication uses a technique that involves a shared secret, which works like a password. Many authentication techniques send passwords with no encryption, allowing them to be compromised by an unauthorized party. However, Kerberos solves this problem via encryption. Instead of sending the password, an encrypted key derived from the password is communicated and thus the password is never sent as plain text. This technique can be used to authenticate a client, but can also be used for mutual authentication of a server. Once authentication takes place, all further traffic is to be encrypted, allowing even new encryption keys to be communicated securely.

Generally, authentication protocols are verified by the use of exact models representing the protocols. Usually, the representative models describe the principals including intruders or attackers. While it may be easier to describe an attacker with simple attributes, it is more difficult to describe attackers with complex attributes in situations where third parties can easily be realised. As Meadows stated in his study entitled, “Formal verification of cryptographic protocols: A survey. In Advances in Cryptology", the current methods of verification adopt formal approaches for code generation and analysis of protocols [25].

In their work, "Local search in model checking", Roscoe *et.al* [24] stated that, A popular mathematical algebra namely CSP developed by [26] has recorded the most remarkable progress in the perspective of describing concurrent systems. FDR is a model checking tool developed by Formal Systems, that establishes results about concurrent and reactive systems described in CSP. FDR functions check if implementation refines the specification of the system. In this research, systems considered are authentication protocols. Despite the unrivalled power of description inherent in the CSP language, the process of producing CSP scripts by hand remains difficult, error prone, and time-consuming. CASPER, a compiler, which translates high level description of protocols to CSP codes by Lowe [11], is progressing and is achieving the required results. Each section of the CASPER script has different tasks, such as declaration of the agents, which are taking part in the protocol, definition of the protocol itself, which runs the protocol [6].

In order to model protocols, the participants in the protocols are modelled as well [3, 14]. In a simple protocol, it is assumed that there are two communicating principals, A and B, and an adversary who is the attacker. The attacker is modelled as having capacity to intercept messages in all directions, modify messages, inject new messages and transmit messages [14].

As stated previously, to present the model of the attacker in CSP, initial steps involve determining the extent of information that could be available to an attacker with formerly mentioned potentials. In a simple protocol, there are two communicating principals, A and B, and an adversary, who is the attacker are represented as [14];

1. with unknown number of clients:

$\text{NET} = (|||j \in_{USER} \text{USER}_j) \mid [\text{trans, rec}] \mid \text{ATTACKER}$

2. with only two participants (client/agent):

$\text{NET} = (\text{USER}_A \mid\mid\mid \text{USER}_B) \mid [\text{trans, rec}] \mid \text{ATTACKER}$

The attacker is modelled in a way to have the capacity of intercepting messages in all directions, modifying messages, injecting new messages and transmitting messages [14]:

$$\text{ATTACKER sat } \left(\text{INIT } \cup (\text{tr } \Downarrow \text{trans})\right) \vdash \text{tr } \Downarrow \text{rec}$$

This theorem is used here to explain that the sets of all the messages that pass through the rec channel are a function of the initial knowledge of the attacker and the sets of the messages input on the trans channel follow:

$$\text{ATTACKER(S)} = \text{ trans?i?j?m} \Rightarrow \text{ATTACKER } (\text{S } \cup \text{m}) \Box \text{ i, j } \in \text{USER, S } \vdash \text{m}$$
$$\text{rec.i!j!m } \Rightarrow \text{ATTACKER (S)}$$

The proposed framework and protocol script [8] provide a design of security solution for wireless local area networks. Since Kerberos is a trusted third party authentication protocol, its paradigms and entities are finalised for the proposed framework [4]. This framework is a timed model security protocol; that uses timestamps to delay of messages by intruders decryption (i.e. attacker cannot break communication for certain amount of time). An approach has been proposed to stop access to authentication server for a period of time to enable the distribution of randomly generated keys to users [6]. Keys are renewed at various intervals while external access to the system is disabled. Another protocol has been developed with the combination of Kerberos Authentication Protocol and Encrypted Key Exchange Protocol [5] with the addition of "delay decryption" property of the Kerberos Authentication Protocol [8].

Apart from these, with the use of the inference rule to analyse a typical Kerberos protocol in the presence of the TGS reveals that the protocol is subject to a TGS masquerade attack. As discussed in the study of [3], authentication in Kerberos requires a **client,** *C*, to send a request to the **authentication server**, *AS*, requesting credentials for a given **application server**, *V*. The *AS* responds with the requested credentials consisting of a ticket and a session key encrypted with the client's key. Kerberos exchanges may also be in the presence of a TGS. The CSP model of inference is as follows:

1. C → AS: Options || ID_C || Realm || ID_{tgs} || Times || $Nonce_1$
2. AS → C: Realm || ID_C || $Ticket_{tgs}$ || $E_{kc}[K_{c,\ tgs}$ || Times || $Nonce_1$ || $Realm_{tgs}$ || $ID_{tgs}]$
3. C → TGS: Options || ID_v || Times || $Nonce_2$ || $Ticket_{tgs}$ || $Authenticator_c^b$
4. TGS → C: $Realm_c$ || ID_c || $Ticket_v$ || $E_{kc,\ tgs}$ $[K_{c,v}$ || Times || $Nonce_2$ || $Realm_v$ || $ID_v]$
5. C → V: Options || $Ticket_v$ || $Authenticator_c^c$
6. V → C: $E_{kc,v}$ $[TGS_2$ || Subkey || Seq ♯]

Nevertheless, the same study shows that, in distributed systems where the intruder has reasonable communication and computational power belonging to the same administrative domain, Kerberos may be compromised. In other words, the chance of impersonating a principal by an intruder is higher where AS and TGS are on the same broadcast network.
Additionally, Trace Semantics are used by [14] to specify security properties, such as secrecy of passwords and the keys, key agreements, for protocols as trace specifications. This is done with the following definitions:

$$\mathrm{P}\ \mathbf{sat}\ \mathbf{S} \Leftrightarrow \forall\, \mathbf{tr} \in \mathrm{traces(P)} \bullet \mathrm{S(tr)}$$

where *P* is a **process** and *S* is a **predicate**. *P* **satisfies** *S*, if *S(tr)* holds for every **trace,** *tr* of P. In terms of occurrence of events in its traces, the following definition is used for some sets of events R and T:

$$\mathrm{P}\ \mathbf{sat}\ \mathrm{R}\ \mathbf{precedes}\ \mathrm{T} \Leftrightarrow \forall\, \mathrm{tr} \in \mathrm{traces(P)} \bullet (\mathrm{tr} \upharpoonright \mathrm{R} \neq <> \Rightarrow tr \upharpoonright T \neq <>)$$

where a **process** *P* **satisfies** the **predicate** *R* **precedes** *T* if any occurrences of an event from *T* is **preceded** by an occurrence of an event from *R* in every **trace,** *tr* of *P*.

As mentioned before, in the same study of [14] and [15], a set of rules is introduced and defined as well to verify the specifications. According to this study, set of rules defined as **atom A**, in this, another three sets are considered which are known as the set of participant identities on the network to be **U,** the set of nonces used by the participants in protocol run as **N** and the set of encryption keys used as **K**. Altogether, the atoms are defined as $\mathbf{A} = \mathbf{U} \cup \mathbf{N} \cup \mathbf{K}$**.** A message space **M** contains all the messages and signals that appear during the protocol's run in a way that $m \in A \Rightarrow m \in M$. A rank function **ρ** is defined in order to map events and messages to integers, $\boldsymbol{\rho}\mathbf{:}\ \boldsymbol{M} \rightarrow \boldsymbol{Z}$. This space is divided into two parts for characterising those messages that the intruder might get hold of [14]:

$$\boldsymbol{M_{P-}} = \boldsymbol{m} \in \boldsymbol{M} \mid \boldsymbol{\rho} \leq \mathbf{0}$$
$$\boldsymbol{M_{P+}} = \boldsymbol{m} \in \boldsymbol{M} \mid \boldsymbol{\rho} > 0$$

where $\boldsymbol{M_{P-}}$ is defined as a non positive rank, for those messages that the enemy should never get hold of, where $\boldsymbol{M_{P+}}$ is assigned for positive rank of, without compromising the protocol.

A general rank function theorem is presented in order to ensure that a protocol will be verified to be correct with regard to its security properties, such as secrecy of passwords and the keys, if all the steps of the theorem are proven [14]. For the sets R and T, there is a rank function, $\boldsymbol{\rho}$**:** $\boldsymbol{M} \rightarrow \boldsymbol{Z}$:

1. $\forall\, m \in \mathbf{IK} \bullet \rho(m) > 0$
2. $\forall\, S \subseteq M.(\rho(S) > 0 \wedge S \vdash m) \Rightarrow \rho(m) > 0$
3. $\rho(B) \leq 0$
4. $\forall\, i.(USER_i|[A]|Stop)\, \mathbf{sat}\, \rho(tr \upharpoonright rec) > 0 \Rightarrow \rho(tr) > 0$

then (|||j $\in_{USER}$ USER$_j$) | [trans, rec] | ATTACKER **sat** R **precedes** T. In condition **4**, the notation $\boldsymbol{tr} \upharpoonright \boldsymbol{rec}$ denotes the projection of trace *tr* onto the channel *rec*: in other words, the subsequence of rec events occurring within *tr*. This requirement on USER$_i$ blocked on a is that if only positive rank messages are received, then no non-positive rank message should be produced. The application of theorem is as follows:

1. Every message in *IK* has positive rank.
2. If every message in a set *S* has positive rank, and $S \vdash m$, then m has positive rank.
3. USER$_B$ does not have positive rank.

3. Work Done

Despite the multiplicity of authentication approaches for improving network security, different types of attacks continue to evolve.

3.1 The Proposed Protocol

The proposed framework [4] has been designed for providing a trusted third party authentication for wireless networks that require a high level of security. Furthermore, the proposed protocol script is based on Kerberos and uses Encrypted Key Exchange Protocol [5]. Since the proposed protocol is a timed model security protocol, it uses timestamps to delaying the decryption of messages by intruders. In a previous study [6], the proposed approach was shutting-down external access to a network for a period of time (e.g. 140 seconds), to enable the distribution of randomly generated keys to users in a relatively secure way. The protocol has been further improved with the introduction of the “delay decryption” property of the Kerberos Authentication Protocol [7]. During protocol verification and validation, it was shown that, due to the strong encryption assignments and authentication specifications there were no attacks found, even when new options “Guessable” and “Crackable” are added to “#Intruder Information” section [8].

3.1.1 The Framework of the Proposed Protocol

The protocol proposed in this paper is a version of the protocol reported in [9] within the designed framework of [8]. In the framework, both the program and data containing the credentials of the legitimate entities of a particular wireless LAN environment are installed on each of the entities as well as TGS and KDC. The credentials are the identities of the devices (such as MAC addresses) and they are stored with cryptographic protection. The protocol adopts the challenge-response paradigm. The interactions between the entities are represented using numbers 1-19 [4]. This is shown in the following **Fig. 1**.

1. The supplicant (wireless client) sends an Extensible Authentication Protocol over LAN (EAPOL) start message to the authenticator (access point) requesting authentication.
2. The access point (AP) responds with a challenge to the supplicant to supply the supplicant's device identity. The AP also bundles the MAC address of the AP itself along with the challenge on actual network traffic under strong encryption to the supplicant.
3. The supplicant responds to the AP after processing the challenge. The supplicant processes the challenge by decrypting the challenge text and ensuring that the AP's MAC address is found in the supplicant's database of possible APs that the supplicant response is also under strong encryption.
4. The AP challenges the authentication server (AS). The challenge text is bundled with the AP's and the supplicant's MAC address still under strong encryption.
5. The AS sends an EAP over LANs (EAPOL) start message to the KDC requesting authentication.
6. The KDC responds with a challenge to the AS to supply the AS's device identity. The KDC point also bundles the MAC address of the AS itself along with the challenge on actual network traffic under strong encryption to the AS.
7. The AS challenges the KDC. The challenge text is bundled with the AS's, AP's and the supplicant's MAC addresses still under strong encryption.
8. The KDC responds to the application server's challenge after processing the content of the challenge text. The process involves the decryption of challenge text and a conformation or proof of knowledge of the existence of both AS and the address of the server. The KDC sends this response under encryption to the AS.
9. The AS sends an EAPOL start message to the TGS requesting authentication.
10. The TGS responds with a challenge to the AS to supply the AS's device identity. The TGS points also bundles the MAC address of the AS itself along with the challenge on actual network traffic under strong encryption to the AS.

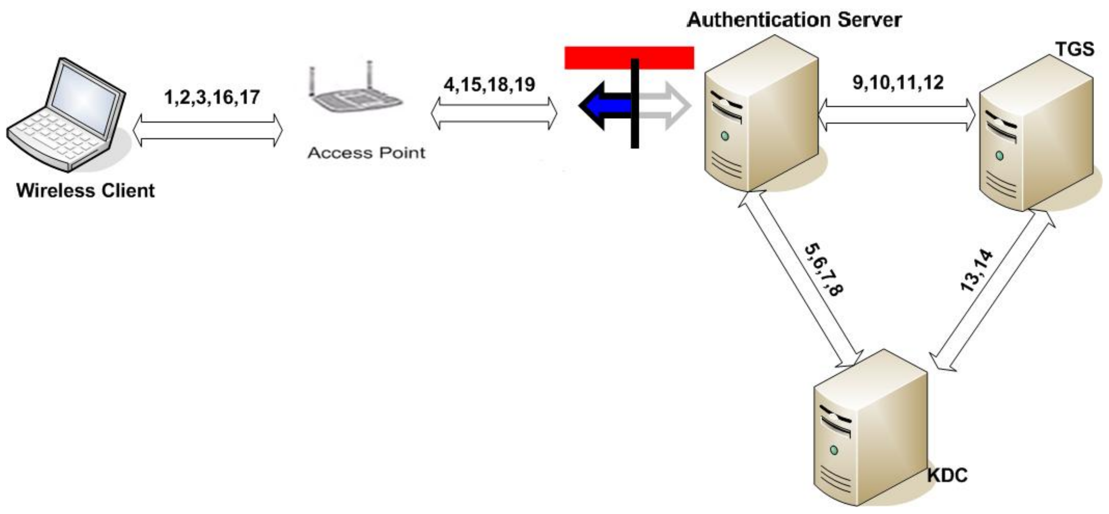

Fig. 1. Proposed Framework

11. The AS challenges the TGS. The challenge text is bundled with the AS's, AP's and the supplicant's MAC addresses still under strong encryption.
12. The TGS responds to the AS's challenge after processing the content of the challenge text. The process involves the decryption of challenge text and a conformation or proof of knowledge of the existence of both AS and the address of the server. The TGS send this response under encryption to the AS.
13. The KDC server challenges the TGS. The challenge text is bundled with the KDC's, AP's and the supplicant's MAC addresses still under strong encryption.
14. The TGS responds to the KDC's challenge after processing the content of the challenge text. The process involves the decryption of challenge text and a conformation or proof of knowledge of the existence of both KDC and the address of the KDC. The TGS sends this response.
15. The AS responds to the AP's challenge after processing the content of the challenge text. The processing involves the decryption of the challenge text and a conformation or proof of knowledge of the existence of both the AP and the supplicant within the secured database of the server. The response includes the MAC address of the server. The AS sends this response under encryption to the AP.
16. The AP challenges the supplicant to run the program to authenticate the end user.
17. If the user responds correctly to the authentication request, the supplicant responds accordingly to the AP.
18. The AP sends the users sign-on response from the supplicant to the AS for the necessary processing.
19. The AS responds to the AP with either the ACCEPT packet or the REJECT packet depending on the outcome of the processing, to the AP. This makes the AP to transition to the authorised state to allow traffic to and from the supplicant with the ACCEPT message or unauthorised state with the REJECT message.

The proposed protocol that has improvements over Kerberos authentication is designed to improve security and minimise the threat of possible attacks. Firstly Kerberos Authentication protocol's capability is checked. New protocol model is designed to minimise possible attacks.

Since it minimises possible attacks, new improvements will be introduced on the specifications and description of the protocol. This is done by the validation of *delay decryption* and *timed authentication* encryption of entities and then, encryption of the messages will be tried with different keys as well as passwords. It has been observed that when these changes on the protocol are checked with CASPER, time to find flaw within the protocol, has significantly increased compared to existing protocol taken as a reference. The results are presented in **Table 1**. It shows that the new designed protocol is superior to previously reported ones in terms of unit times for finding flaw within the protocol [8].

Table 1. Comparison of time to find flaw within the protocol

Protocol	Times (secs.)
Needham-Schroeder	68
Kerberos Key Exchange without delay-decryption	140
Kerberos Key Exchange with delay-decryption	220
Kerberos Key Exchange with delay-decryption and password encrypted	360

3.2 The CSP Model of the Proposed Protocol

In this section, the CSP representation of the proposed protocol is modelled as a network. CSP scripts are designed by depending on appropriate sections of the protocol script. While modelling the different processes of a protocol, advantage of the extensibility of CSP gives the opportunity to add additional elements to the processes. The following scripts are representations of three participants of the proposed protocol under *#Processes* section:

```
#Processes
INITIATOR(A,S,na) knows SK(A), SPK, SKey(A),
PK(A), passwd(A,B)
RESPONDER(B,S,nb) knows SK(B), SPK, SKey(B),
passwd(A,B)
SERVER(S,kab) knows PK, SSK(S), SKey, passwd

INITIATOR(A, S, na) =
[] B : Agent @ A != B & env_I.A.(Env0, B,<>) ->
output.A.S.(Msg1, Encrypt.(SKey(A), <B>),<>) ->
[] kab : SessionKey @ [] ts : TS @ [] pkb :
addGarbage_(PublicKey) @
input.S.A.(Msg2, Encrypt.(passwd(A, B), <S,
A, Timestamp.ts,
Encrypt.(PK(A), <kab>), pkb>),<>) ->
output.A.B.(Msg3, Encrypt.(passwd(A, B), <A,
Timestamp.ts, na, Encrypt.(pkb, <kab>)>),<>) ->
[] nb : Nonce @ [] tb : TS @
input.B.A.(Msg6, Encrypt.(inverse(kab),
<nb, na, Timestamp.tb>),<>) ->
[] ta : TS @ output.A.B.(Msg7, Encrypt.(kab,
<nb, Timestamp.ta>),<kab>) ->
close.A.INITIATOR_role -> STOP
```

In the #Processes" section, the first parameter of each *process* (here A, B and S) should represent agent identities used in the #Protocol description" section. Names are given to the roles played by the different agents. In this protocol, INITIATOR, RESPONDER, and SERVER are chosen. The parenthesized parameters and the variables following the keyword “knows” define the knowledge that the agent is expected to have at the beginning of the protocol run.

```
RESPONDER(B, S, nb) =
[] A : Agent @
[] v : addGarbage_({Encrypt.(passwd(A, B),
<A,Timestamp.ts, na, Encrypt.(pkb, <kab>)>)
| A <- Agent,B <- Agent,
kab <- SessionKey, na <- Nonce,
ts <- TS, pkb <- addGarbage_(PublicKey)}) @
A != B & input.A.B.(Msg3, v,<>) ->
output.B.S.(Msg4, Encrypt.(SKey(B), <A>),<>) ->
[] pka : addGarbage_(PublicKey) @ [] now : TS @
decryptable(v, pka) and nth(decrypt(v,pka), 1) == A
and nth(decrypt(v,pka), 2) == now and
decryptable(nth(decrypt(v,pka), 3), passwd(A,B))
and decryptable(nth(decrypt(v,pka), 4), SK(B)) &
input.S.B.(Msg5, Encrypt.(passwd(A, B),
<S, B, pka>),<Timestamp.now>) ->
RESPONDER_0'(B, S, nb, A, v, pka,
nth(decrypt(nth(decrypt(v,pka),3))))

RESPONDER'(B, S, nb, A, v, pka, na) =
RESPONDER''(B, S, nb, A, v, pka, na,
nth(decrypt(nth(decrypt(v,pka),4),SK(B)),1))
RESPONDER''(B, S, nb, A, v, pka, na, kab) =
[] tb : TS @
output.B.A.(Msg6, Encrypt.(kab, <nb, na,
Timestamp.tb>),<kab>) -> [] ta : TS @
input.A.B.(Msg7, Encrypt.(inverse(kab),
<nb, Timestamp.ta>),<kab>) ->
close.B.RESPONDER_role -> STOP
```

In this proposed protocol, the initiator *A* and *S* are expected to know own **identity** *A*, *S,* the **nonce** *na*, *nb* and *ns*, the **public key** *PK*, own **secret key** *SK(A)*, **server key** *SKey(A)* and common **password** *passwd(A,B)*. Since, actual work of *S* is designed a SERVER, it is not necessary to define its keys in both INITIATOR and RESPONDER.

```
SERVER(S, kab) =
[] A : Agent @ [] B : Agent @
input.A.S.(Msg1, Encrypt.(SKey(A), <B>),<>) ->
[] ts : TS @
output.S.A.(Msg2, Encrypt.(passwd(A, B), <S, A,
Timestamp.ts, Encrypt.(PK(A), <kab>), PK(B)>),<>) ->
input.B.S.(Msg4, Encrypt.(SKey(B), <A>),<>) ->
output.S.B.(Msg5, Encrypt.(passwd(A, B),
<S, B, PK(A)>),<>) ->
close.S.SERVER_role -> STOP
```

In the model above, keywords *input* and *output* are used to define *receive* and *send* application respectively, where *trans* and *rec* keywords are the general definition for this purpose in Schneider's CSP definitions. Also, instead of key words USER$_A$, USER$_B$, *INITIATOR* and *RESPONDER* are used. The CSP representations of each process (known as agents) show that the entities of the messages are encrypted with their own public keys.

```
#Protocol description
0. -> A : B
[A != B]
1. A -> S : {B}{SKey(A)}
2. S -> A : {S, A, ts, {kab}{PK(A)}, PK(B) \
% pkb}{passwd(A,B)}
3. A -> B : {A, ts, na, {kab}{pkb \
% PK(B)}}{passwd(A,B)} % v
[A != B]
4. B -> S : {A}{SKey(B)}
5. S -> B : {S, B, PK(A) % pka}{passwd(A,B)}
[decryptable(v, pka) and nth(decrypt(v,pka), 1) == A \
and nth(decrypt(v,pka), 2) == now \
and decryptable(nth(decrypt(v,pka), 3), passwd(A,B)) \
and decryptable(nth(decrypt(v,pka), 4), SK(B))]
<na := nth (decrypt (nth(decrypt(v,pka), 3))) ; \
kab := nth (decrypt (nth(decrypt(v,pka), 4), SK(B)), 1)>
6. B -> A : {nb,na,tb}{kab}
7. A -> B : {nb,ta}{kab}
```

As it is seen from the above script, the protocol is designed in a way that, the INITIATOR, A creates and sends a message, Msg 3, but the RESPONDER, B stores this message in variable v, without trying to interpret it. That is to say, RESPONDER, B decrypts this message and performs the appropriate checks only after receiving message in the future steps, which is defined Msg 5. Msg 3 was encrypted with the inverse of the key received in Msg 5. Since the inverse of the password received is itself, B expects the common password.

```
A != B & input.A.B.(Msg3, v,<>) ->
output.B.S.(Msg4, Encrypt.(SKey(B), <A>),<>) ->
[] pka : addGarbage_(PublicKey) @ [] now : TS
```

In the test of Msg 5, *decryptable,decrypt* and *nth(_,n)* functions are used for delaying decryption purposes. Due to the use of *delay decryption*, B cannot automatically extract any fields from Msg 3. The assignments are added by using the functions encapsulated between < >. The first assignment assigns the **nonce**, *na* as the third field of Message 3, but the message itself, is encrypted with the common *passwd(A,B)*, which is distributed by the server, has to be decrypted using the inverse of this key which is itself. The second assignment assigns the **session key**, kab as the fourth component of Message 3 but the first field of the message is encrypted with B's public key.

```
RESPONDER'(B, S, nb, A, v, pka,
nth(decrypt(nth(decrypt(v,pka),3))))
RESPONDER'(B, S, nb, A, v, pka, na) =
RESPONDER''(B, S, nb, A, v, pka, na,
nth(decrypt(nth(decrypt(v,pka),4),SK(B)),1))
```

The first assignment, RESPONDER', assigns the **nonce**, *na*, as the third field of Msg 3, but the message itself, is encrypted with the **password** of A and B, has to be decrypted using the inverse of this key which is itself. The second assignment,
RESPONDER'(B,S,nb,A,v,pka,na) = RESPONDER''(B,.) assigns the **session key**, *kab* as the fourth component of Msg 3 but the first field of the message is encrypted with B's public key and decryption has to be done by using the inverse of this key which is secret key of B.

```
#Specification
Secret(A, kab, [B])
Secret(B, kab, [A])
TimedAgreement(B, A, 2, [kab])
```

The following CSP process implies that A was running the protocol with B within the last 2 time units and there is mutual agreement between them. This mutual agreement depends on the value of session key chosen:

```
AuthenticateRESPONDERToINITIATOR_TimedAgreement2_kab_0(B) =
addTime(signal.Running1.RESPONDER_role.B?A?kab ->
signal.Commit1.INITIATOR_role.A.B.kab -> STOP,2)
AlphaAuthenticateRESPONDERToINITIATOR_TimedAgreement2_kab_0(B)
=
union({|signal.Running1.RESPONDER_role.B.A,
signal.Commit1.INITIATOR_role.A.B |
A <- inter(Agent, HONEST)|},{tock})
```

In addition to the *timestamps*, *nonces* are also being used for authentication between the agents A and B. The intruder can try to make the first attempt to attack in Msg 3 when A sends **nonce**, *na* to B. Due to protocol *time agreement* specification, an agent’s chance to attempt to connect will be timed-out by the server because of unsuccessful connection attempts thus preventing the attack.

Additionally, in the #Intruder Information section, the intruder’s identity and the set of data values that he knows are initially stated. The *IntruderKnowledge* section holds the identifiers and functions of the protocol that he knows and he can apply any other value to those identifiers and functions:

```
#Intruder Information
Intruder = Mallory
IntruderKnowledge = {Alice, Bob, Mallory, Sam, Nm, PK,
SPK, SK(Mallory), SKey(Mallory)}
Guessable = SessionKey
Crackable = SessionKey
Crackable = ServerKey
Crackable = Password
```

Intruder has an initial knowledge of *Guessable* values and entities. Referring to the following script, the intruder closes up its initial knowledge under deductions and calculates the *fact* that cannot be learnt.

The CSP definition for the above script is:

```
IK0_init = union({Alice, Bob, Mallory, Sam, Nm, SK(Mallory),
SKey(Mallory),
Garbage}, TimeStamp)
INTRUDER_1 = (chase(INTRUDER_0)
[[ hear.m_ <- send.A_.B_.(l_,m_,se_) |
(l_,m_,se_,re_) <- DIRECT_MSG,
A_ <- diff(SenderType(l_),{Mallory}), B_ <-
ReceiverType(l_) ]] [|{| hear |}|] STOP)
[[ say.m_ <- receive.A_.B_.(l_,m_,re_) |
(l_,m_,se_,re_) <- DIRECT_MSG, A_ <- SenderType(l_),
B_ <- ReceiverType(l_) ]]
```

The job of Intruder is defined in the above code together with its relevance to its initial knowledge. The initially **known** *facts* are added under the following section. The key words such as *leak*, *hear*, *say*, are used to signal that a possible secret has been learnt, to represent hearing and saying a message during the authentication and transfer of a message across a network.

The keyword *leak* is used to signal that a possible secret has been learnt. Elements such as ***f_*** , ***IK1*** are components of the intruder for currently unkown *fact* and *initial knowledge* respectively.

```
SAY_KNOWN_0 =
(inter(IK1, ALL_SECRETS_DI) != {} &
dummy_leak -> SAY_KNOWN_0)
[] dummy_send -> SAY_KNOWN_0
[] dummy_receive -> SAY_KNOWN_0
SAY_KNOWN = SAY_KNOWN_0
[[ dummy_leak <- leak.f_ | f_ <- inter
(IK1, ALL_SECRETS_DI) ]]
[[ dummy_send <- send.A_.B_.(l_,m_,se_) |
(l_,m_,se_,re_) <- DIRECT_MSG, components_(m_)
<= IK1, A_ <- diff(SenderType(l_),{Mallory}),
B_ <- ReceiverType(l_) ]]
[[ dummy_receive <- receive.A_.B_.(l_,m_,re_) |
(l_,m_,se_,re_) <- DIRECT_MSG, components_(m_)
<= IK1, A_ <- SenderType(l_),
B_ <- ReceiverType(l_) ]]
STOP_SET = {| send.Mallory |}
IntruderInterface = Union({{| send, receive |}, {|crack|}})
AlphaSystem = {|env, send, receive, close, tock|}
SystemManagerInterface = inter(AlphaSystem,
CRACKING_M::AlphaManager)
SYSTEM = (SYSTEM_M::TSYSTEM_1
[|SystemManagerInterface|]CRACKING_M::Manager)
[|IntruderInterface|] INTRUDER_M::INTRUDER
```

3.3 Constructing Rank Functions

In this section the rank functions of the NET on the message space are constructed [13, 14]. The following steps and table show the rules that are constructed while creating rank functions.

1. All user **ID**s in the set ***U*** are assigned to a positive rank: It is assumed that **ID**s are known to Intruder and can be impersonated.
2. All the nonces in ***N*** are assigned to a positive rank: It is also assumed that Nonces are known to Intruder Knowledge.
3. In the protocol there are three different keys *SessionKey* which is defined as *kab*, *ServerKey* as *SKey* and Password as *passwd*. *kab*, *passwd* are assigned to a non-positive rank, because they are supposed to be private to agents in the protocol.
However, *SKey* is assigned to a positive rank, with the same rule that is mentioned in previous two steps.
4. The running messages between input and output channels are assigned to a non-positive rank.
5. This step came along with the previous step, which is related with signal events
Commit, *Running* where input and output channels respectively.

The summary of these assumptions is as follows:

Table 2. Rank function for the protocol

Step	Rank function
1	ρ(U) = 1
2	ρ(N) = 1
3	ρ(K) = { 0 if: kab or passwd else: 1 (SKey)
4	$\rho(m_K)$ = {0 if: m(kab) or m(passwd) else: 1 (m(SKey))
5	**ρ**(A,B,kab,na) = 1 and **ρ**(B,A,nb, na) = 0

The rank function theorem is defined in terms of general sets **R, T**. In this research, **R** and T are assigned to the step 5 of above table and it is extended as:

Initiator - output = R = ρ(Running,A,B,kab,na) = 1
Responder - input = T = ρ(Commit,B,A,nb, na) = 0

Since the rank functions are constructed as provided above (Table 2), the conditions of the rank function theorem should be checked in order to make sure that the functions provided above are satisfied.

1. $\forall\, m \in IK \bullet \rho(m) > 0$
In protocol description IK is Intruder Knowledge and at the beginning it contains all users in the NET and nonce of itself, where all the contents are positive rank. Therefore, the set is positive rank. The condition is deemed satisfied.

2. $\forall\, S \subseteq M, m \in M \bullet \big((\forall\, m' \in S \bullet \rho(m') > 0) \wedge S \vdash m\big) \Rightarrow \rho(m) > 0$
This condition checks if whether a message of a non-positive rank can be generated from a given set of messages of positive rank. Messages are non-positive rank(step 4 of table 2). Intruder generates any messages that are non-positive rank. These messages are messages of steps 3, 5 and 6 at protocol description, which are generated by passwd, kab. Both are non-positive rank. This prevents the Intruder from generating these keys. Therefore no way of getting messages. This condition is satisfied.

3. $\forall\, t \in T \bullet \rho(t) \leq 0$
In this condition, all events in T are non-positive rank. Since **Responder - input** = **ρ**(B,A,nb, na) = **ρ**(Commit,B,A,nb, na) is non-positive rank. This condition is satisfied.

4. $\forall\, i \in U \bullet USER_i \underset{R}{\|}$ **Stop** maintains ρ
Every process in NET needs to maintain positive **ρ.** However, events are restricted in set R. **Initiator - output** = **ρ**(A,B,kab,na) = 1 = R. A, and B are restricted on **Initiator - output**. Therefore it will be checked if they maintain positive **ρ**.

$$USER_A \;\|\; Stop = \,\square_b$$
$$\textbf{Initiator} - \textbf{output} = (\mathrm{A}, \mathrm{B}, \mathrm{na})$$

$\square_b$ is choice operation and b indicates the other participants (B and S). Therefore it satisfies to maintain positive **ρ**.
The protocol is checked and proven to be successful in improving security against attacks despite of strong intruder connection attempts. This is demonstrated in steps given in this section.

4. Conclusion

This paper presents the CSP codes and the construction of rank functions of a new protocol proposed for improving security of the Kerberos Authentication Protocol. CSP Processes and rank functions constructed, give opportunities of better understanding of security protocols and focuses on relevant design aspects of these protocols. The new protocol is an enhanced version of a previously reported protocol [6, 7] designed to increase the time for an intruder to find flaw within the protocol and hence improve security.

Schneider's CSP Processes [14] and rank function theorem [13, 14] are applied to expose any flaws in the design and possible attacks have been successfully identified. Analysis shows that the proposed protocol has achieved the goal of increasing the time needed to find flaw within the protocol, and hence improve security. In addition to “delayed decryption”, this paper proposes further improvement to network security by restricting access to the Kerberos authentication server temporarily, i.e. during the process of key exchanges. While CSP shows protocol vulnerabilities, it has its limitations. The use of rank functions further improves protocol validation through tracing intruders’ capabilities, and knowledge. Through the use of rank functions, this paper gives a more accurate validation of the proposed protocol highlighting security improvements achieved.

Last but not the least, this research aims to develop authentication frameworks for the design and implementation of effective and efficient mechanisms for Kerberos authentication protocol used in wireless communication networks. The design of authentication protocols spans well over three decades with the foundation of authentication protocols by [12]. While Kerberos approach has been proposed as a standard for enhanced security in IEEE 802.11i Task Group on Security (TGi) [23], currently there are no valid proposals using a Kerberos-like mechanism to provide authentication in a wireless network, preventing from cryptographic attacks and handling fast and secure handovers. In this research, authentication protocols for the IEEE 802.11 networks, based on the IEEE 802.11i works are proposed and Kerberos protocol is used to provide a framework for these purposes.

This work describes original research on development of security strategies using Kerberos in wireless communication networks. Due to the critical nature of wireless communication networks, the existing methods are insufficient to address perspective and requirements in authentication design. Since Kerberos authentication protocol is considered as a solution, various existing methods and solution techniques for Kerberos, its basic operation in wireless communication networks are described, studied and analysed. The timed authentication protocols, with delaying decryption properties are proven to improve the time for a potential attacker to find a flaw within the protocol.

The proposed framework is to the best of our knowledge the first study which combines delayed decryption and temporary interruption to link/server access for improved security. Rank function theorem is usually applied to validate and expose compromises of the existing protocols in the literature, such as stated in Shaikh and Bush [15]. In this research paper although analysis of the proposed protocol is discussed and presented by CSP, and analysis tool CASPER, its verification is proved by the Rank function theorem as well. Timed delayed and temporary link interruption properties are successfully validated against strong intruder model. The rank function theorem employed is a well-accepted and detailed method that involves the search of all states reachable by the components of the proposed model. Since the proposed approach involves temporary interruption to link/server access, it has implications in terms of performance degradation. Work is in progress in using an analytical method to evaluate the cost in terms of the degradation of system performance. The model being developed will consider the system for exact performability evaluation.

References

[1] M. Abadi, and R. Needham, “Prudent Engineering Practice for Cryptographic Protocols,” *IEEE Trans. Softw. Eng.*, vol.22, pp.6-15, 1996. Article (CrossRef Link)

[2] V. Cortier, J. Delaitre, and S. Delaune, “Safely Composing Security Protocols,” *Formal Methods in System Design*, vol.34, no.1, pp.1-36, 2009. Article (CrossRef Link)

[3] A. Eneh, O. Gemikonakli, and R. Comley, “Security of Electronic Commerce Authentication Protocols in Economically Deprived Communities,” in *Proc. of The 5th Security Conference 06*, Las Vegas, Nevada, April 2006, ISBN: 0-9772107-2-3.

[4] Y. Kirsal, A. Eneh, and O. Gemikonakli, “A Solution to the Problem of Trusted 3rd Party of IEEE

802.11b Networks," in *Proc. of 6th Annual Postgraduate Symposium (PGNET 2005)*, pp.333-339, 2005.

[5] Y. Kirsal, and O. Gemikonakli, "An Authentication Protocol to Address the Problem of the Trusted 3rd Party," *Authentication Protocols Innovative Algorithms and Techniques in Automation, Industrial Electronics and Telecommunications*, pp.523-526, Springer, Netherlands, 2006. Article (CrossRef Link)

[6] Y. Kirsal, and O. Gemikonakli, "Frequent Key Renewal Under Pseudo-Secure Conditions For Increased Security in Kerberos Authentication and its Impact on System Performability," in *Proc. of Proceedings of the 3rd International Conference on Global E-Security*, University of East London (UeL), 2007.

[7] Y. Kirsal, and O. Gemikonakli, "Further Improvements to the Kerberos Timed Authentication Protocol," *Novel Algorithms and Techniques in Telecommunications, Automation and Industrial Electronics*, pp.550-554, Springer Netherlands, 2007. Article (CrossRef Link)

[8] Y. Kirsal, and O. Gemikonakli, "Improving Kerberos Security through the Combined Use of the Timed Authentication Protocol and Frequent Key Renewal," in *Proc. of 7th IEEE International Conference on Cybernetic Intelligent Systems (CIS2008)*, pp.153-158, IEEE Press, 2008. Article (CrossRef Link)

[9] Y. Kirsal, and O. Gemikonakli, "Analysing the Kerberos Timed Authentication Protocol Using CSP-Rank Functions," in *Proc. of the 5th International Conference on Global Security, Safety and Sustainability, ICGS3'09*, pp. 56-63, 2009. Article (CrossRef Link)

[10] G. Lowe, "Some New Attacks upon Security Protocols," in *Proc. of 9th IEEE Computer Security Workshops*, pp.162-169, Society Press, 1996. Article (CrossRef Link)

[11] G. Lowe, "Casper: A Compiler for the Analysis of Security Protocols," in *Proc. of 10th Computer Security Foundations Workshop (CSFW '97)*, vol.18, no.30, IEEE Computer Society, 1997. Article (CrossRef Link)

[12] R. M. Needham, and M. D. Schroeder, "Using Encryption for Authentication in Large Networks of Computers," *Communication ACM*, vol.21, pp.993-99, ACM Press, 1978. Article (CrossRef Link)

[13] A. W. Roscoe, "CSP and Determinism in Security Modelling", *In Proc. IEEE Symposium on Security and Privacy*, pp.114-127, Society Press, 1995. Article (CrossRef Link)

[14] S. Schneider, "Verifying Authentication Protocols in CSP," *IEEE Trans. Software Eng.*, vol.24, pp.741-758, IEEE Press, 1998. Article (CrossRef Link)

[15] S. Shaikh, and V. Bush, "Analysing the Woo-Lam Protocol using CSP and Rank Functions," *Journal of Research and Practice in Information Technology*, vol.38, pp.19- 29, 2006. Article (CrossRef Link)

[16] A. Roy, A. Datta, A., Derek, J.C. Mitchell, and J. P. Seifert, " Secrecy Analysis in Protocol Composition Logic," *Advances in Computer Science (ASIAN 2006), Secure Software and Related Issues Lecture Notes in Computer Science*, vol. 4435, no. 2, pp 197-213, 2006. Article (CrossRef Link)

[17] A. Datta, A. Derek, J.C. Mitchell, and A. Roy, "Protocol Composition Logic PCL," *Electronic Notes in Theoretical Computer Science*, vol. 172, pp. 311 – 358, 2007. Article (CrossRef Link)

[18] SECWP. Security White Paper Evolution, Requirements, and Options. Technical report, Symbol Technologies Inc., 2007.

[19] S. Zrelli, and Y. Shinoda, "Specifying Kerberos over EAP: Towards an Integrated Network Access and Kerberos Single sign-on Process," in *Proc. of International Conference on Advanced Information Networking and Applications*, IEEE ComputerSociety, 2007. Article (CrossRef Link)

[20] J. Kohl, and C. Neuman. The Kerberos Network Authentication Service (v5). RFC, MIT/ The Internet Society, United States, 1993.

[21] C. Neuman, and T. Ts'o, "Kerberos: An Authentication Service for Computer Networks," *IEEE Communications Magazine*, vol. 32, no.9, : pp. 33-38, September 1994. Article (CrossRef Link)

[22] J.F. Monin, "Understanding Formal Methods," Springer-Verlag New York, Inc., Secaucus, NJ, 2001. ISBN 1852332476.

[23] M. A. Kâafar, L. B. Azzouz, F. Kamoun, and D. Males, "A Kerberos-Based Authentication

Architecture for Wireless LANs," *NETWORKING*, pp. 1344-1353, 2004. Article (CrossRef Link)

[24] A. W. Roscoe, P. J. Armstrong, and Pragyesh, "Local Search in Model Checking," in *Proc. of the 7th International Symposium on Automated Technology for Verification and Analysis, ATVA'09*, pp. 22-38, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-04760-2. Article (CrossRef Link)

[25] C. A. Meadows, "Formal verification of cryptographic protocols: A survey," *Advances in Cryptology (ASIACRYPT'94)*, vol. 917 of Lecture Notes in Computer Science, pp. 133-150, Springer Berlin / Heidelberg, 1995. Article (CrossRef Link)

[26] C.A. R. Hoare, "Communicating sequential processes," *Communications of the ACM*, vol. 21, pp:666-677, 1985. Article (CrossRef Link)

Yoney Kirsal-Ever obtained her BSc. degree from the Department of Computer Engineering, Eastern Mediterranean University, Cyprus in 2002, her MSc in Internet Computing from University of Surrey, Guilford, Surrey, UK in 2003, and her PhD from School of Engineering and Information Sciences in Middlesex University, London, UK. Also, in 2012 she completed her Post Graduate Certificate in Higher Education, in Middlesex University as well. Her research is on development of security strategies using Kerberos in wireless networks. Yoney has worked as a part-time lecturer while she was doing her BSc and PhD. Currently; she is a technical tutor at Middlesex University, London. She has fully refereed and published international conference papers with various awards including IEEE best paper for promising research. Her research interest is in network security, authentication protocols and formal verification methods. Dr. Kirsal-Ever has been a Member of ACM since 2007 and a Member (M) of IEEE since 1998. Also, she was one of the team members of Eastern Mediterranean University, IEEE CSIDC2002 Competition.

Dr. Agozie Eneh is a senior lecturer in the Department of Computer Science, University of Nigeria. He has just finished serving as the Head of Department of Computer Science in September, 2014. Dr Agozie Eneh holds a PhD in Computer Network Security obtained from Middlesex University London in 2007. Other degrees include; MSc Computer Networks, MBA in Management, and a BSc in Computer Science. He has worked in both public and private sectors. Prior to joining the University of Nigeria, Dr Agozie Eneh was special assistant (Technical) to the ministers of Defence and to the minister of Agriculture in Nigeria. He worked for Computer Systems Associates Lagos from 1995 to 1998 before joining Softscience Technologies Ltd where he still holds the position of lead consultant. Dr Eneh's current research interests include; authentication protocols analysis, network security, optimisation theories, medical informatics, and assistive technologies for educating children and adolescents with learning difficulties.

Orhan Gemikonakli was born in Cyprus in 1958. After graduating from Lefke Gazi Lisesi, he studied BSc Electrical Engineering at Middle East Technical University. He received his BEng Electrical Engineering from Eastern Mediterranean University in 1984. He then continued his studies at Kings College, University of London in UK obtaining MSc in Digital Electronics, Computers, and Communications, and PhD in Telecommunications in 1985 and 1990 respectively where he worked as a Postdoctoral Research Associate from January 1989 to August 1990. He Joined Middlesex University (then a Polytechnic) in September 1990 as a Lecturer. He worked there as a Senior Lecturer, Principal Lecturer and PG Curriculum Leader. After serving as the Acting Academic Group Chair (January-July 2007), he was appointed to the post of Head of Department of Computer Communications in the School of Engineering and Information Sciences at the same University. Currently, he is the Head of Department of Computer and Communication Engineering, School of Technology and Sciences. Prof Gemikonakli has extensive teaching experience in telecommunications, signal processing, computer networks, operating systems, object oriented programming, and network security. His research covers telecommunications, performability evaluation of complex systems, modelling and simulation, and network security. He serves on various conference Programme Committees and editorial boards.

Leonardo Mostarda is an Associate Professor at University of Camerino. He is currently the head of the department of computer science. His research interests include various areas of distributed systems and security. He received his computer science Ph.D. from the University of L'Aquila in 2007. In 2007 he was research associate at Imperial College London and in 2010 Senior Lecturer at Middlesex University, London. He participated to several European, Italian and UK projects where he had the opportunity to collaborate with several institutions such as European Space Agency and Ministry of Cultural Heritage.