AUTHENTICATION MECHANISM ENHANCEMENT UTILISING SECURE REPOSITORY FOR PASSWORDLESS HANDSHAKE

Ioannis A. Pikrammenos, Panagiotis Tolis and Panagiotis Perakis

School of Computing, Mediterranean College – University of Derby, Athens, Greece

ABSTRACT

In this paper the idea of an enhanced security authentication procedure is presented. This procedure prohibits the transmission of the user's password over the network while still providing the same authentication service. To achieve that, Kerberos Protocol and a secure password repository are adopted, namely a smart card. The conditional access to a smart card system provides a secure place to keep credentials safe. Then, by referencing to them through identifiers, an authentication system may perform its scope without revealing the secrets at all. This elevates the trustworthiness of the mechanism while at the same time it achieves to reduce the overhead of the authentication systems due to the elaborate encryptions procedures.

KEYWORDS

Kerberos v5, LDAP, authentication, password handling, smart card

1. INTRODUCTION

In this paper a methodology is introduced, capable to prevent the exposure of the user's password during the authentication phase in a network. The methodology focuses on well-known authentication mechanisms and provides a stepwise approach of security performance improvement. The method achieves to utilize a set of pre-computed passwords hidden to anyone but the authentication system. The method identifies passwords through their position in a list and uses them in random order. The change of passwords infrequent intervals leaves little room for attackers while the system-oriented generation and usage of passwords (transparent to the user) allow for security best practices to be applied. The enhanced capabilities of password utilization improve the performance of the systems as the cryptographic functions are not necessarily applied in a repeatable manner. Several technologies have been reviewed in order to compose a solution available for both open source and proprietary operating systems.

The paper is divided as follows; the first section consists of a literature review followed by a section analysing the authentication protocols. Next section embodies the security features of each proposal. The last section presents the proposed solution and its main outcomes. A discussion about challenges and future work are followed by conclusions.

1.1. Authentication and identification

Authentication comes from the Greek lemma *αὐθεντικός* (authentikos) which means "the act of confirming the truth of an attribute of a single piece of data claimed to be true by an entity". Such attributes indicate the identity of the data by profiling it through its characteristics. Identification refers to the act of stating/indicating a claim purportedly attesting to an entity's identity, namely a person or thing's identity**.** Authentication is the process of validating that identity.

To comprehend the role of user identification and authorization into a computer system we should take an insight into its operational logic. A computer system is designed and developed to manage and coordinate the resources of available processors, memories, storage areas, and attached peripheral devices. These resources are utilized for diverse purposes, consisting of unique combinations known as processes or instances. It is clear that if multiple instances try to utilize the same resources at the same time, then the computer system would not be capable to serve them. To cope with that, certain prioritization is developed among instances to avoid a system crash. According to this logic, an instance has greater priority over another instance according to its criticality. Because the computer system cannot decide for all instances on this basis, humans do. And now a question arises: which among computer system's users is the decisive one? Any user on a computer system wants to have the maximum priority on his choices, status that is about to be changed if we consider system-specific critical operations (i.e. as heat exhaustion) or sharing the computer system with other users, as such users should be divided into categories according to their decisive role over the system operation.

The prime user category is the system's administrator that has the role of configuring the operation of the system and fine-tuning the provided services. Another category is the simple use of the system's offered services that has a decisive role only on the services offered. Finally, a role called guest is introduced, satisfying the overwhelming need for a temporal user that shares very limited resources of the system.

User identification was introduced to allow the system to identify the user and thus to specify his access privileges. In early times, the user was identified through the effective user id (euid), used for all purposes. It was a code name that represented the user to the system. This code name, called username or user id, was a symbolic and not an actual description of the human user: the username neither had any significance in the real world nor should have any relevance to the user's description (name, surname, etc).

1.2. Username

For a human to interact with the computer, he should use the common interfaces, usually the keyboard and the screen. As such, the username should be created using the system-wide character set. The system's character set incorporates letters as well as numbers and symbols that come from human language. Still, there are more "symbols" that may extend the communication pattern from human-centric to system-specific. For example, the *carriage return* (ASCII character 13 CR) has no significance in human communication but has a severe impact on system processes. These system-specific symbols (control characters) may not be used in usernames as their result would be devastating for the proper operation of the system. In some cases, language-specific characters belonging to the extended character set (something that is not standard among systems) are not taken into account for usernames.

Utilizing the system-specific character set as described above someone could form combinations of his wish as usernames. Still, there is the system's operational design that implies shape and structure limitations. In structure, the username should not be separated into discrete words (sets of characters) by using the special character space (ASCII character 32). In shape, the username should meet a specific limit of characters. The upper limit depends on the system, for example in Windows 7 it is 20 characters. The lower limit is one (1) character as there shouldn't be a "no one" (from Homer Odyssey) user, even though the symbolic value for username class "nonexistent" is -2.

With time, computer systems become more complicated offering diverse services. To maintain the proper operation of these services as well as to update them without interfering with other services and resources, a specific class of username had to be introduced. That username class would be system-centric, as the operations required are limited in authorization and very narrow in extent. As such, usernames were distinguished according to the system they were about to be used, namely “fuid” for a file system, “ruid” for remote login on border equipment and so on.

Every system realizes a realm where procedures and resource sharing are maintained in order. Multiple systems have diverse requirements and restrictions, and when brought together to interoperate, extend of systems-specific discipline applied is leveraged. System interoperation imposes the exploitation of common requirements and the abolition of the diverse ones. This means that if a user wants to use identical usernames among multiple systems, the codeword used in the former system could be prohibited in the latter. If there is no standard applied to impose the common criteria, username restrictions are not the same among systems and each system should maintain its own usernames. A user interacting with multiple systems should keep handy multiple usernames to log on each of them separately. This is a burden that most users could handle for a handful usernames. Given the increase to the number of usernames or their complexity, users find it difficult to remember them, failing to access system resources properly.

The diversity of usernames that a single user must keep up is a security issue by itself. It is so because it is not feasible to keep this information in mind. Usernames cannot be possessed by a single user and also should be unique to each system applied. Users should select usernames that do not match those of existing user accounts and comply to system requirements. That increases the complexity of a codeword, usually driven away from the naming habits of the user. Usernames that are not familiar to user habits or comprehension may not be memorized easily. Commonly, humans note complex usernames on paper to keep track of them. Having the username on a media that is accessed easily, meaning that policing the paper reading is not easily applied, anyone without proper permission could gain knowledge of the written username. Given the fact that users gain access to systems based on usernames, the third unauthorized party has achieved half the way in almost effortlessly.

Another security drawback is that a username once created is fixed, unchanged. Computer systems link usernames with the storage area where user-specific information is stored. This means that the username as codeword is correlated with the folder name used for storage. If the username codeword was to be altered, then either the folder name should be changed, or the system operation should be altered to link username to a folder name that is not identical. The prior is not feasible, as once created by the administrator, the contents of the “user named” folder belong solely to the user. The latter is also not feasible, as systems’ operation is predetermined. A third party that happens to know the exact combination of system characters that form a username, or even reveals it, could potentially claim it. The only defense the original user has is to delete the account and create a new one with a different username. Still, there is no way to keep the username away from third parties as that would dissolve the meaning of remote service provision.

1.3. Password

User authentication is a process that follows user identification. Given the drawback of username’s uniqueness into a system, along with the necessity to share user identities among services, a process that would reassure the system for the authenticity of the part that uses the username is required. For this to happen a communication mechanism should be established so as the user to prove his authenticity to the system. Systems interact with the user through standardized interfaces, and for this reason, the password is engaged. Password is another code word that is used for user authentication. It is linked to a username and identifies uniquely the owner of it. Alike the username codeword, password uses a character set that is valid for the occasion. In this case, it incorporates all the character set along with more symbols than the username does.

The purpose of the password is to be kept secret, shared only among user and system, but has the same disadvantages as usernames. Its intend is to harden the possibility to be revealed. Despite that, the more secure a password is the more difficult it becomes for humans to conceal it or even remember it.

As the unique technique to verify user's authenticity, passwords became a single point of security failure. Malicious users try to break passwords with several techniques in order to gain access into systems. Those techniques (such as brute force, dictionary, etc) took advantage of authentication methods and system operations vulnerabilities.

- The first one is the transmission of the password "over the air" meaning that anybody that has access to the transmission medium could "listen" (eavesdrop) to the password. For this reason, passwords were encrypted. Encryption transforms the original character string to a string of bytes. This transformation brings passwords' value beyond the valid character set and as such become non-comprehensible to anyone that does not know the encryption procedure. Still, given that systems should interoperate, and as such encryption techniques should be well established, someone could reveal the password through elaborate methods of reverse encryption (decryption). To cope with that, a procedure that introduced arbitrarily to the encryption process was employed (the so-known "salt") hardening the decryption. Though tough, decryption is still possible [1].

- A second drawback is that password codeword is also exposed by its shape, meaning the length of the codeword in characters. This knowledge limits the effort taken to break the password, and the shorter the password is the faster to be broken. To cope with this, the passwords should be enhanced to maintain their meaning but to be length neutralized. This enhancement allows for the uniformity in password lengths as well as concealment for their specific characteristics. The homogenous alteration of passwords was feasible with the hash algorithms, like MD5. The hash algorithm has a security advantage that it can not be used in the opposite direction, meaning that a hashed password cannot be decrypted. Still, new attacks arose, like the rainbow one [2].

- The protection of the communication channel among interacting parties, the user and the system, became a necessity. Encryption algorithms, like SHA256, were developed to guarantee channel confidentiality. When the remote parties were synchronized, the channel becomes practically non-penetrable, providing the required password protection. Still, if someone achieved to play the roles of the remote party (client or server) and stand in the middle of the communication channel, he could misguide process through replaying messages from the legitimate parties and in time decrypt content [3].

- The man-in-the-middle attack gave advent to mutual authentication procedures, like PKI, where a participant in a secure channel conversation should be recognized as valid before and during this act. For this to happen, a set of authentication and encryption procedures take place with the utilization of the private and public key. This method has the limitation that the keys are fixed for each user and so, exposed in time and attacks, vulnerable.

- To provide a secure space where even the keys could be changed in time so as the procedure to be sanitized and the content remain safe, complicated systems like Kerberos where developed. In such a system not only all of the aforementioned procedures take place, but it is also foreseen the periodical change of keys and passwords. The attack space is limited due to often changes and penetration is extremely difficult, but not impossible. What keeps penetration possible is the transmission of the passwords and keys over “the air”.

A complete solution to security issues over authentication mechanisms would be *the avoidance of transmission of any password or key over the air*. The usage of these codewords in procedures like challenges could trigger trust relationship and provide a leeway for securing channel without an observer ever see the codeword in over the air. This solution is proposed as an extension to the current functionality of the Kerberos system through the usage of a secure repository, the smart card, as described further in this paper.

2. AUTHENTICATION MECHANISMS

Authentication indicates how one party verifies another's identity. Can be one-way or two-way (the latter is sometimes called mutual authentication). In one-way authentication, a client presents a password to a server, or a server presents its certificate to a client during an SSL/TLS connection negotiation so that the connection can be encrypted. In two-way authentication, both the client and server exchange certificates during SSL/TLS connection negotiation. Besides the certificate exchange, on any authentication procedure, it is mandatory for a client to present its credentials in order to gain access to the service. Usually, these credentials are transmitted over the network via an encrypted channel. That channel usually becomes a target from malicious users focusing to gain the credentials transferred within.

Several technologies have been used to create a security proposal available on both open-sources (Lightweight Directory Access Protocol, Kerberos) services. The following section consists of a review over the mandatory device, the protocols, and their functionality. Furthermore, the security aspect of the protocols will be emphasized, as security is the key issue of this paper.

2.1. Lightweight Directory Access Protocol

LDAP (Lightweight Directory Access Protocol) is a standard specified by RFC 4510. It is a "light" application of X.500 targeting directory services. The main service of LDAP is to distribute information among clients connected on the system. LDAP refers to a set of 4 models that guide the client as he browses a directory [4]:

1) The **naming model** describes the organization and reference to the data. Data are organized in logical entities in a hierarchical manner, building up structures, the directories. Addressing specific block is achieved with naming.

2) The **information model** describes the data types storable in folders as well as the structure of information units.

3) The **functional model** consists of a set of operations divided into three groups. Among them, the authentication and control operations allow clients to identify themselves to the directory and control certain aspects of a session.

4) The **security model** relies on the fact that LDAP is a connection-oriented protocol. In other words, an LDAP client opens a connection to an LDAP server and performs various protocol

operations on the same connection. During the lifetime of the session, the LDAP client may authenticate to the directory server. At that point, supplementary privileges may be given. Furthermore, as mentioned above the fact that LDAP is a connection-oriented protocol creates the necessity of multiple security enhancements.

It is mandatory for the client to authenticate to the directory server multiple times depending on the policies applied over the network. With the term policies, a number of variables are taking into consideration, such as the lifetime of the session (timestamps) and privileges are given by the administrator. Authentication from the client's perspective is the process of proving to the server that the client is a particular entity by proving it with a username and a password. From the server's perspective, the process of authentication involves accepting the identity and credentials provided by the client and checking whether they prove that the client is who it claims to be.

2.1.1. Security Over LDAP

By providing a Domain Name (DN) and a set of credentials, a client can use the **bind operation** to authenticate itself to the directory. The server checks if the credentials of the specified DN are legitimate. Then, the server notes that the client is authenticated for the connection lifetime.

The server then grants privileges to the client on the basis of its identity. Client identity reaches the server through bind methods. There are different types of bind methods in LDAP. In a simple bind (Figure 1), the user presents a clear-text DN and password to the LDAP server. The server verifies that the user's password matches the value stored in the entry's user password attribute and, if so, gives the client a success code.

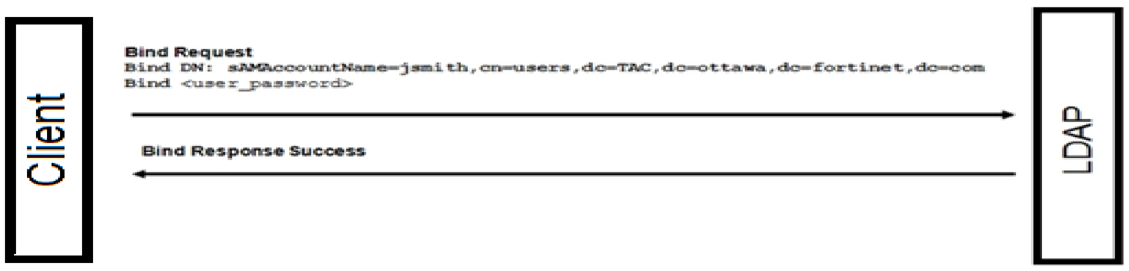

Client
Bind Request
Bind DN: sAMAccountName=jsmith,cn=users,dc=TAC,dc=ottawa,dc=fortinet,dc=com
Bind <user_password>
Bind Response Success
LDAP

Figure 1. LDAP simple Bind authentication

In version 3 of LDAP a new type of bind operation is introduced: The Simple Authentication and Security Layer (SASL) bind. With SASL, the client indicates the sort of protocol to be used (SSL, TLS, DIGEST-MD5). If the server supports the authentication protocol, an agreed-on protocol is performed between the client and server [4]. In this way, a password is not transmitted in plaintext.

2.1.2. Threats Against LDAP

In March 2014, researchers found vulnerability (Heartbleed) in OpenSSL, the cryptographic library used to secure connections. This permitted connection endpoints to scan packet data after payload by denoting a length of payload greater than the amount of data expected in the HeartbeatRequest message. The Heartbeat Extension enables either endpoint of a TLS session to identify whether its peer is still active and was driven by the need of Datagram TLS (DTLS) session management. Standard TLS applications do not contain the extension because they can depend on TCP to manage the corresponding session. During the original TLS handshake, peers show approval for the expansion. Either endpoint may send a HeartbeatRequest signal after negotiation to check connectivity. HeartbeatRequest posts are a one-byte type field, a two-byte payload field, a payload field and at least 16 bytes of random padding [5]. A solution against the Heartbleed vulnerability is a mechanism to check that the response is not longer than the request.

While buffer overhead or overflow attacks can be patched, session hijacking attacks need more drastic countermeasures. Man-in-the-Middle (MitM) [6] attack relates to "a type of active wiretapping attack in which the intruder intercepts and deliberately modifies transmitted information to mask as one or more of the persons engaged involved in a communication association". The attack will be analyzed more in the next section, as it affects Kerberos as well via the use of SSL/TLS from the protocol. SSL/TLS Protocols are vulnerable to MitM as their authentication mechanisms can be bypassed if the attacker manages to make his attack on the real-time environment [7]. Based on the nature of LDAP and the usage of SSL/TLS, a MitM attack becomes a threat that cannot be ignored.

2.1.3. Vulnerabilities of LDAP

The harassment of the communication channel among LDAP client and server imposes severe security threats to the authentication service. Still, the service can be restored while the incident cannot. The incident reflects the exposure of the service to malicious handling, namely the exposure of the secrets. In our case, secrets are the password of the client transmitted over the channel for the authentication procedure. It is so because the rest of the user credentials, like username and domain name, are dominated by the operation of LDAP (naming model, etc) that is inflexible. Thus, the main drawback of LDAP is its lack to preserve password confidentiality if channel integrity is breached.

2.2. Kerberos

Kerberos offers a means of validating the identity of individuals on an accessible (unprotected) network (e.g., a workstation operator or a network server). This is achieved without depending on host operating system authentication, nor relying on host addresses or forcing all hosts on the network for physical security. Under these circumstances, Kerberos conducts authentication by using standard cryptography, i.e. shared a secret key, as a trusted third-party authentication provider.

Sharing a secret requires multiple parties, so a shared DES key is a secret key. Something is private only when it is not known to anyone but its proprietor. Thus, a sole entity keeps both public and private keys in public-key cryptosystems.

2.2.1. Authentication Procedure

Kerberos originally was built based on symmetric-key cryptography and requires a trusted third party. Extensions to Kerberos can provide for the use of public-key cryptography during certain phases of authentication. The client needs to be authenticated to the resource server and requests a session key from the KDC. The KDC will produce and distribute the session key to both parties. It must transmit it to both the client and the resource server after the KDC has produced the session key. Kerberos encrypts it with the entity's master key to ensure the transmission of the session key to a specific entity. Thus, it is necessary to generate two encrypted versions of the session key: one is encrypted with the master key of the client, and the other is encrypted with the resource server's master key. The session key encrypted with the master key of the resource server is regarded as a "ticket" in Kerberos terminology. A Kerberos ticket offers a manner to safely carry a Kerberos session key across the network. Only the destination resource server decrypts it [8]. Kerberos authentication relies on a procedure implemented with the exchange of messages as described below (Figure 2):

- **AS_Request:** The client presents the workstation with a username and password. Additionally, he provides a query to issue a ticket-granting-ticket (TGT) for the ticket-granting system (TGS) for the authentication server (AS). It involves the username but not the password in plaintext. On the registered password, the client conducts a one-way hash function, and this becomes the client's hidden master key. The client sends the request to the AS.

- **AS_Reply:** When the AS accepts the TGT application, it extracts the username from the request and retrieves from its internal database the respective password and produces the client's master key by hashing the password. The AS then creates a TGT, wrapping the TGT in a response message. The TGT includes both plaintext and an encrypted piece. To encrypt the ciphertext portion of the TGT, the AS utilizes a cryptographic key extracted from the user's password. Thus, the encrypted part, which also contains a cryptographic key known as a session key, can only be decrypted by the user who knows the password. The AS sends the response message to the requesting client.

- **TGS_Request:** The client receives the reply transmission, extracts the TGT, then decrypts the encrypted portion of TGT. Furthermore, the client presents a service ticket request. The request will wrap the TGT and an authenticator-known encrypted framework. Using the session key obtained from the TGT, the client encrypts the authenticator. The authenticator verifies the awareness of the session key of the client. The request for the service ticket also indicates the resource server name. The client sends the service ticket request to the TGS.

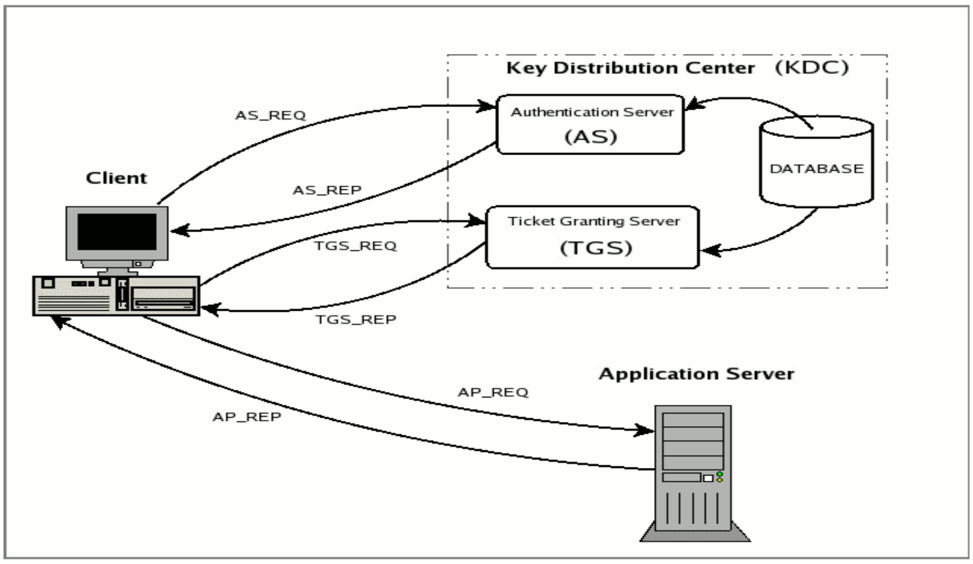


Figure 2. Kerberos authenticated Service Provision Architecture

- **TGS_Reply**: The TGS extracts the name of the server for which the client requests the service ticket upon receipt of the service ticket request. Then, a service ticket is composed. The TGS encrypts the service ticket's ciphertext portion with the server's secret key so that this portion can only be decrypted by the server. Also, the TGS involves a fresh cryptographic key called a sub-session key in the ciphertext portion of the service ticket. The TGS returns to the client the response message.

- **APP_Request**: When the TGS reply is received, the client uses the session key to decrypt its ciphertext portion to obtain the sub-session key. The client also extracts the service ticket. Then, the client authors a message for the resource server and wraps the service ticket in the message. This message represents a request to the server for establishing a new secure session with the client. The client sends the message to the resource server.

- **APP_Reply**: The resource server derives from the request the service ticket, decrypts its ciphertext section, and then collects the session key. Thus, both the client and the server will be aware of this key. Next, the resource server grants the client a successful affirmation. With this session key, the client and resource server can now interact safely with each other.

2.2.2. Security Features

Within Kerberos scope is the prevention of unencrypted password handling. As such the user's password is stored in AS database in encrypted form utilizing the string2key function. This function is called every time a new password enters the realm of a principal. The string2key function is employed to transform a diverse formed user password in a suitable uniform secret key. The function utilizes the PBKDF2 mechanism to perform its purpose. Password-Based Key Derivation Function version 2, (PBKDF2), is a key derivative function released in PKCS #5 by RSA Laboratories. PBKDF2 introduces CPU intensive activities to deal with brute force attacks based on weak user passwords. Such activities are based on a pseudorandom iterated (PRF) feature. PBKDF2 uses a pseudorandom function, for instance, a hash-based message authentication code (HMAC), including a password (P) together with salt (S) as entries and replaying (N) amount of times to create a secret key (K). Hash-based Message Authentication Code (HMAC) is an algorithm based on a cryptographic hash function to generate a message authentication code. The HMAC definition needs

- H: a selected cryptographic hash function,
- K: the secret key
- text: a message to authenticate

As described in RFC 2104, HMAC can be defined as follows:

$$HMAC = H(K \oplus opad, H(K \oplus ipad, text))$$

Salt is a string to be concatenated to the unencrypted password before applying the string2key function to obtain the key. Kerberos 5 utilizes the same user's principal as salt:

Ktolis = string2key (Ptolis + "tolis@EXAMPLE.COM")

Ktolis is the encryption key of the user tolis and Ptolis is the unencrypted password of the user. This type of salt has the following advantages:

1. Two principals belonging to the same domain still have distinct keys with the same unencryp ted password. For example, imagine an administrator having a principal for everyday work (tolis @ EXAMPLE.COM) and one for administrative work (tolis/admin @ EXAMPLE.COM). This User has likely set the same password for both principals for reasons of convenience. The salt presence ensures that the associated keys are distinct.
2. If a user has two accounts in different domains, it is fairly common for both domains to have the same unencrypted password: due to the presence of the salt, a possible compromise of an account in one realm does not automatically lead in the other being breached. [9].

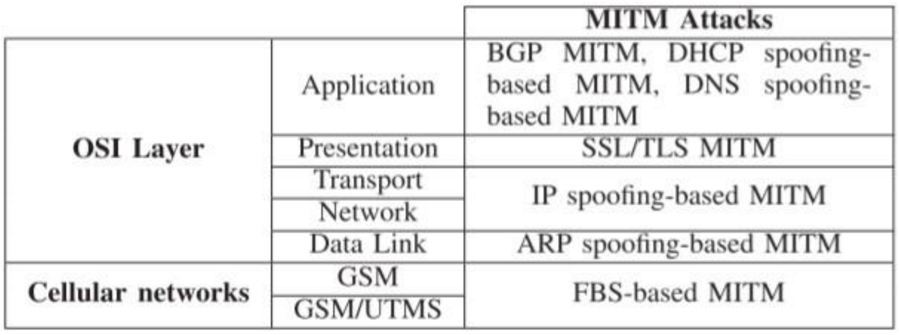
By default, Kerberos uses as salt principals' name and the name of the realm. The absence of a salt input to the hash is opposed to cryptographic best practices and enables an attacker to develop a password hash "rainbow table."

2.2.3. Threats Against Kerberos

Given the poor-quality of the user-selected passwords, a rainbow table derived from common passwords likely list would be able to compromise Kerberos principals in any realm (for example) using RC4 encryption types for password-derived keys. While multiple proposals for enhancing the cryptographic aspect of Kerberos exist, numerous threats can bypass that type of augmentation. Rainbow tables are a compromise between precomputation and low memory usage, in essence "saving memory at the cost of cryptanalysis time." About to the potential issue, the KDC must remember the secret key of every user in the conventional Kerberos setting. Therefore, if an illegitimate person even gains read-only access to the KDC database, the KDC's security is thoroughly compromised. Alternatively, since Kerberos utilizes public key enabled keys to encrypt TGTs, read-only access to the database of a KDC would not compromise security at all. The only way for an intruder to breach the public key-enabled security of Kerberos is to acquire write access to the folder of the X.509 Certificate Authority. This is considerably more difficult than reading secret keys passively in traditional Kerberos databases [8]. Thus, implementing any type of cryptography would upgrade the security infrastructure of Kerberos, the protocol remains vulnerable against multiple types of attacks.

Man-In-The-Middle (MitM) is a kind of attack where a malicious third party secretly takes control of the communication channel between two or more endpoints. The attacker of MitM can intercept, modify, alter or substitute the communication traffic of target victims. Besides, victims are unaware of the intruder, believing that the channel of communication is protected. While MitM can be executed in different communication channels (Table 1), our focus is over SSL/TLS as those two protocols are in charge of securing the channels between the client and the server. Furthermore, SSL / TLS MitM is a method of active network interference in which the intruder embeds himself into the medium of communication between two victims (generally speaking, the browser of the victim and the webserver). The intruder then creates two distinct SSL links with each victim and relays data between them in such a manner that the middleman is unaware of both. This configuration allows the attacker to record all data on the wire and even alter the transmitted data selectively [10].

Table 1. MitM Attack on Different Layers of OSI Model and Types of Networks [10]



		MITM Attacks
OSI Layer	Application	BGP MITM, DHCP spoofing-based MITM, DNS spoofing-based MITM
	Presentation	SSL/TLS MITM
	Transport	IP spoofing-based MITM
	Network	
	Data Link	ARP spoofing-based MITM
Cellular networks	GSM	FBS-based MITM
	GSM/UTMS	

2.2.4. Vulnerabilities of Kerberos

Standard Kerberos protocol weakness is that the key kC used to encrypt the client's credentials is derived from a password, and passwords are notoriously vulnerable to dictionary attacks. Besides, since the initial request is completely plaintext, an active attacker can repetitively request credentials from a truthful client and accumulate a large amount of plain text ciphertext pairs, the latter being encrypted with the long-term key kC of the client. While the attacker cannot use these credentials to authenticate the scheme, there is a significant chance for the attacker to carry out an active dictionary attack against the key [11].

3. Proposition decomposition

As discussed in the previous sections, the main drawback of authentication mechanisms is the transmission of the password over the channel. Even hardened through the encryption of the transmission channel, the exposure of the secret is still possible. To surpass this drawback, a method that prohibits the exposure of the secret over the air is necessary. Relevant works elevate the security characteristics of the communication channel that secret is passed over, but not the method of secret sharing.

Secrets precomputed and stored in a secure repository can be delivered to remote users without the usage of the communication channel. The secure repository is a user limited replica of the authentication authority repository decentralized on a smart media. This media is transferred under the control of the user but not under his authority. This repository shall contain secrets that only the authenticating authority knows. To produce diversity, the secrets shall be listed in sets and identified through their row number. The valid password is identified through the list number and used locally, keeping it away from transmission over the channel. A third-party eavesdropping channel traffic can comprehend the reference (secret number) but not the content. After the sealing of the channel through cryptography, all unauthorized parties are locked out. Until they achieve a breakthrough of current password seal, an alteration of the valid password may take effect without the exposure of its value.

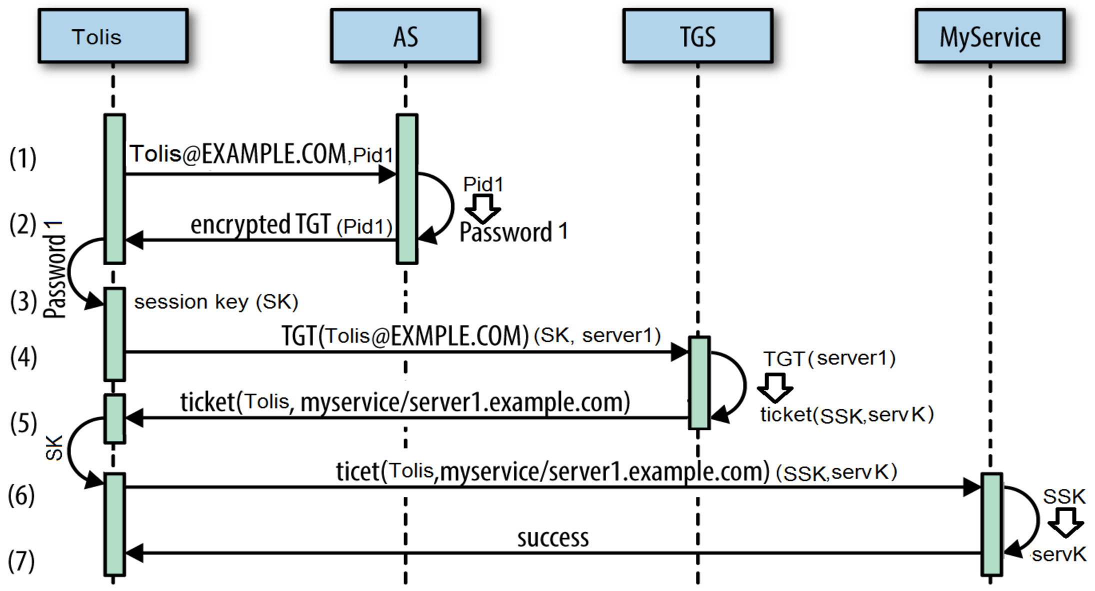


Figure 3. Kerberos standard authentication procedure

The proposed scheme incorporates the smart card as a securer credentials' repository. Smart cards may control the access to their file system while at the same time separate the file systems in isolated areas. As such, each user possessing a smart card may hold diverse sets of credentials per required authentication procedure (i.e. remote access and back-office access). The valid credentials consist of the current username and password as opposed by the authentication authority through the namely procedure. Credentials' current values are deterministic for the authentication to happen though stochastic in time. This means that their value is valid until the user alters them. Alteration allows for a new password, yet the username cannot follow that procedure as services provided to user from authentication authority are linked to that (username). A set of passwords are linked to each username to change the credentials set dynamically. Through this method, issuer and user are synchronized regarding the authentication credentials required through a given set of passwords pre-computed and linked to the given username by the issuing authority.

Usernames are fixed for the life of use in a system, but not necessarily communicated to the user. This means that as the username, as an authentication credential, participates in the above-mentioned authentication procedure, its current value may be kept away from the user knowledge. As it is stored in the smart card file system, it may be accessed by the authentication authority when required and kept secret elsewhere. This prohibits the user from unintentional disclosure of the half part of the credentials' secret, the username.

As the authentication procedure may choose a set of credentials through the potential combinations of the pre-computed passwords, there is no need for password renewal with the intervention of user or any other system outside the secure repository environment of the smart card. Given the order of passwords enlisted, one could identify the credentials set through the identifier of the selected password's thesis along with the pre-computed list. In this way, no one except the issuer-authentication service may know the actual value of the identified password. The lack of password knowledge of the user reduces the chance of human factor security issues on a domain. In this way, security considerations mentioned at (RFC 4120) as the secrecy of principals' keys, password-guessing attacks and so on are raised.

The password is known only to the system, not the user, and as such, it cannot be exposed. As the user does not have to enter the password, and namely to remember it, the password length and complexity could be arbitrary, leaving space for security best practices to be applied from scratch. Then passwords are pre-selected to be strong enough to apply as countermeasures for the aforementioned recognized Kerberos' implementation and cryptographic problems. It is necessary to pick a compatible with Kerberos cryptographic algorithm even though the smart card will not work as a cryptographic device but as storage. Currently, Kerberos v5 supports smart card implementations. The proposed solution is taking advantage of the architecture of microprocessor smart cards. Also, a smart card comprises a computer system that retains secrecy as it only becomes operational when plugged into a suitable hosting device while maintaining data in various security layers at the same moment. Their file system (Figure 4) contains multiple partitions creating the finest structure for the proposal, as each partition can be programmed to require authentication and store different passwords individually for each host (smart card per host) [12]

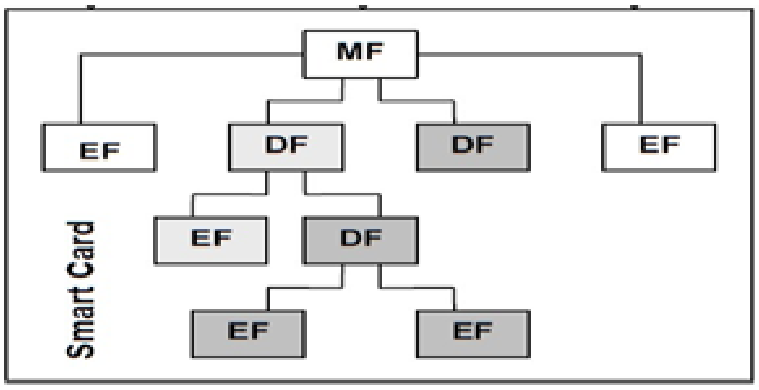


Figure 4. Smart Card File System /DF: Dedicated File / MF: Master File / EF: Elementary File

3.1. Proposed Solution Implementation

3.1.1. Kerberos Oriented Implementation

1) The source code of Authority Service Request step needs to be changed to receive a different type of credentials (besides the user's name, a smart card's ID code is mandatory as well as the enlisted password id):

- AS_REQ = (PrincipalClient, PrincipalService, IP_list, Lifetime) is the existing code and it requires the above information.

The new version needs to contain additional information in the form of (Table 2):

- AS_REQ = (PrincipalClient, PrincipalService, IP_list, Lifetime, **SCidCode, Pid**)

2) Database scheme at the AS has to transform the password record from a fixed size variable to a fixed size list. The list may hold all of the alternative pre-defined passwords in a hierarchical order to be accessed sequentially. A password would be then selected according to its turn in raw (1st, 2nd, etc.). Also, it is necessary to add new tables and entities in the database such as smart card's serial number including the partition numbers that the passwords are stored, to cover every possible new variable necessary for the implementation.

Table 2. Kerberos database schema before and after proposed enhancement.

dn: uid=krb5-kdc,ou=dso,dc=example,dc=com objectClass: top objectClass: account objectClass: simpleSecurityObject uid: krb5-kdc userPassword: {SSHA}OFFICE123456789HAD description: LDAP account for the Kerberos KD	dn: uid=krb5-kdc,ou=dso,dc=example,dc=com objectClass: top objectClass: account objectClass: simpleSecurityObject uid: krb5-kdc **userPassword: SCidCode(Pid)** description: LDAP account for the Kerberos KD **objectClass: SCidCode** **userPassword1: {SSHA}OFFICE123456789HAD** **userPassword2: {SSHA}HAD1234567THISSEAT** **description: Password list in repository**

3) This turn-based identification mechanism could be utilized when a password has to be changed due to changes in applied policy or to life expectancy. As such, no new password is needed to be entered to the system by the user, but just the identification of the raw number of the next applicable pre-stored one (Figure 5). This means that string2key mechanism has not to be invoked on-demand and the system computational resources could be spared.

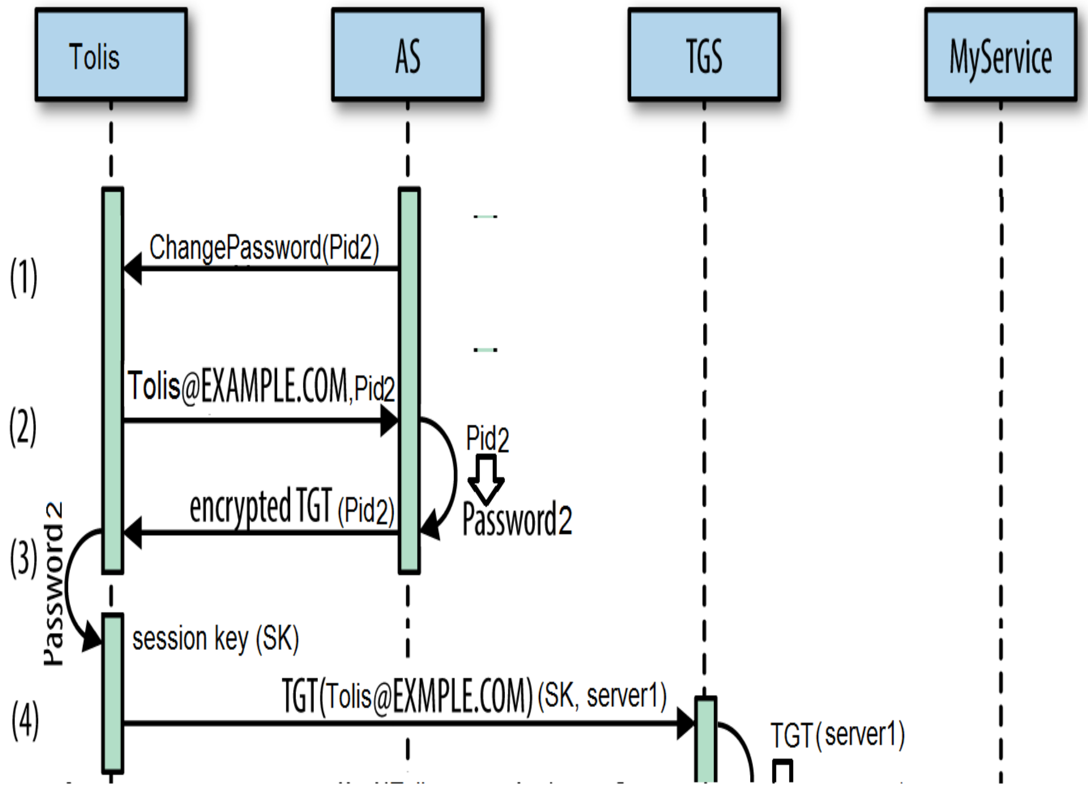


Figure 5. Passwordless handshake.

4) Given the fact that the passwords may now be selected by the system, as there is no more human interaction required, the scope of using string2key and namely PBKDF2 mechanisms is raised. The system password complexity policy provides security guarantees. For example, the password =lw5=sw8praju9O98bu_p_6#?D2at2&ph!0@epld5us9ch@=lp$#Uqu-rimiJ=sP as proposed by Norton Password Generator has about 8064 combinations that are feasible to be hacked in about 2□10111 years . Nevertheless, to keep up with best practices and backward compatibility, the encryption mechanism may remain under the following condition: the variable N of PBKDF2 mechanism must be reduced to a minimum to minimize resources occupation, for instance, match the passwords' id (raw number). This modification makes an intruder's life harder as he still has to guess the exact number of encryption iterations i to expose the original password (of unknown length and uncommon combinations).

5) As the host machine does not contain a local repository (the smart card stores the password), Ticket Granting Server needs to redirect the ticket's destination from the machine to the card. This feature can effectively help mobile users connecting through different devices with the same ticket, reducing the computational power of authenticating the same users every time they change devices. It is mandatory to take into account that the ticket given from Kerberos has an expiration date, given that the host will still have to reauthenticate after the pass of the preselected time.

3.1.2. LDAP Oriented Implementation

The proposal is taking advantage of the usage of Kerberos as an authentication service for LDAP and raises a drawback as it prevents any password exchange over the network. The password may be encrypted with a common secret that AS and user (smart card) hold, like the smart card ID. The password then can be sent on the wire. Upon arrival to the AS, a set of derivatives over the user's smart card ID and the potential passwords are produced to be compared to the incoming encrypted password. Upon a match, the AS validates whether the password is correct and updates the password ID counter. This counter may be utilized for dynamic password changes. The following modifications are necessary to be deployed over the host's machine:

1) Smart card ID should be stored in AS repository and linked to the user's principal name. LDAPv3 supports smart card implementations, giving us the chance to simulate the proposed solution in a prepared environment.

2) The password repository should be changed from the terminal's file system to the smart card. When a smart card is accessed, the user must be invoked to approve the action through entering the PIN.

3.2. Eliminated Threats

Numerous threats against each technology were analyzed in previous chapters. The suggested proposal can work as a countermeasure for the majority of them. Namely:

1. **Man in the Middle attack**: Assuming the attacker has managed to bypass multiple layers of security features, the cryptographic algorithms compatible with Kerberos are complex enough to require a huge amount of processing power. Even if the attacker manages to steal any type of information by hijacking the session, it will not have any value to him as he will have no clue of the secret.

2. **Dictionary – Brute force attacks**: This type of attack focuses on the user credentials in order to hack them. As there is no fixed length or format of the password as users do not interfere with it, there is no use of a dictionary for an attacker. It is possible only to try all the potential space and structure combinations, making the breaking too harsh and, in conjunction with channel security measures, even impossible.

3. **Clone attack**: The possibility to replay legitimate messages to hijack a session is not further handful as the dynamic change of passwords that authenticate users as well as sessions through keys expose such attacks.

4. **Social Engineering**: Key aspect of the proposed solution is the lack of password knowledge from the user's perspective. The user will not be able to expose something that he/she does not know.

4. DISCUSSION

4.1 Related Work

While the majority of current security proposals based on Kerberos protocol are related to the cryptographic aspect of Kerberos [13], [14], a smart card integration design was proposed by [15]. Their design was taking advantage of microprocessor cards by transferring the cryptographic processing from machine to the card. Furthermore, they implemented the following changes to swap the encryption-decryption mechanism from terminal to smart card:

1. New encryption system (DES in CBC with MD5) as a compatibility issue existed between CRC (default hash method of Kerberos v5-1.0.5) and their chosen smartcard.
2. Modified DES library so as Kerberos (that by defaults calls the encryption function from a terminal) search for the smartcard for operation performance.

3. Modified the authentication function so that no inquires are issued for the password from the client (use) but are retrieved from smartcard instead.

The above resolve the issues created by dictionary attacks to the Kerberos authentication procedure, but still, the method remains vulnerable against brute force and Man in the Middle attacks. The encrypted password can be further processed by the attacker if he establishes a real-time session hijack.

4.2. Threats Against Smart Card Implementations

Cryptographic devices have several physical and logical interfaces. Some of these interfaces can be accessed easily, while others can only be accessed by special equipment. Based on the interface that is used for an attack, it is possible to distinguish between invasive, semi-invasive, and non-invasive attacks. All of these attacks can be either passive or active. If the smart card as a secure repository is breached, then the secrets are exposed. As such, care should be taken as the repository be utilized only when meant to be. Future work could focus on securing the authentication process end-to-end.

5. CONCLUSIONS

The proposed solution is implemented in authentication mechanisms, like LDAP and Kerberos. Through the overview of their functionality, a leeway of enhancements that implement the aforementioned solution is presented. The changes required in protocols are minimal while the impact is great. The authentication procedure is further hardened, freed from known vulnerabilities.

REFERENCES

[1] Fabian Monrose, Michael K. Reiter, Susanne Wetzel. (2002) "Password hardening based on keystroke dynamics," International Journal of Information Systems, Volume 1, Issue 2, pp 69–83

[2] Himanshu Kumar, Sudhanshu Kumar, Remya Joseph, Dhananjay Kumar, Sunil Kumar Shrinarayan Singh, Praveen Kumar, Himanshu Kumar. (2013) "Rainbow table to crack password using MD5 hashing algorithm," IEEE Conference on Information & Communication Technologies

[3] R. Bird, I. Gopal, A. Herzberg, P.A. Janson, S. Kutten, R. Molva, M. Yung. (1993) "Systematic design of a family of attack-resistant authentication protocols," IEEE Journal on Selected Areas in Communications, Volume: 11, Issue: 5.

[4] Timothy A. Howes Ph.D., Mark C. Smith, G. S. G. (2003) epdf. Second Edi. Addison Wesley.

[5] Carvalho, M. et al. (2014) 'Heartbleed 101', IEEE Security and Privacy, 12(4), pp. 63–67. doi: 10.1109/MSP.2014.66.

[6] R. Shirey, "Internet Security Glossary", RFC 2828, May 2000

[7] Oppliger, R. et al. (2008) ‘SSL/TLS Session-Aware User Authentication’, (March).

[8] Al-janabi, S. T. F. and Rasheed, M. A. (2011) ‘Public-Key Cryptography Enabled Kerberos Authentication’, 2011 Developments in E-systems Engineering. IEEE, pp. 209–214. doi: 10.1109/DeSE.2011.16.

[9] Ricciardi, F. (2007). KERBEROS PROTOCOL TUTORIAL. [online] Available at: http://editions-internetparc-files.pbworks.com/f/KERBEROS_PROTOCOL_TUTORIAL_Fulvio_Ricciardi.pdf [Accessed 3 June 2019].

[10] Conti, M, Dragoni, N & Lesyk, V 2016, 'A Survey of Man in the Middle Attacks' IEEE Communications Surveys and Tutorials, vol. 18, no. 3, pp. 2027-2051

[11] Scedrov, A. (2008) ‘FORMAL ANALYSIS OF THE KERBEROS AUTHENTICATION Supervisor of Dissertation Acknowledgments’.

[12] Pikrammenos I, Anagnostopoulos V. (2003) “Bidirectional, Multi-Layer Peer-to-Peer Authentication in WLAN Network Using Smart Cards as Mediators,” 8th WSEAS on Communications.

[13] Her-Tyan YEH. (2006) “Improvement of an Efficient and Practical Solution to Remote Authentication: Smart Card,” IEICE TRANS. COMMUN., VOL.E89–B, NO.1

[14] Tanmoy Maitra, Mohammad S. Obaidat, Ruhul Ami, SK Hafizul Islam, Shehzad Ashraf Chaudhry, Debasis Giri. (2016) “A robust ElGamal-based password-authentication protocol using smart card for client-server communication,” International Jounal on Communication Systems 2016; vol 1–12

[15] Itoi, N. and Honeyman, P. (1999). “Smartcard Integration with Kerberos V5. USENIX,” Workshop on Smartcard Technology.

AUTHORS

Ioannis A. Pikrammenos, Dr. Ing., MBA, has graduated Electrical Engineering and Computer Technology at Patras University, Greece and has acquired his doctoral thesis as well as MBA from National Technical University of Athens, Greece. He has been employed as lecturer and researcher on ICT for over 20 years in numerous EU universities and institutions, participated in more than 30 research initiatives, published more than 20 scientific articles or books/chapters and has more than 8,500 teaching hours (EQF 7-6-5). Pioneering in the area of entrepreneurship cohesion has supervised more than 100 business plans and 1 patent. He is member of Greek Technical Chamber (Tee, eTee) and ASHRAE. Research interests: NetWorks, protocols, data and system security, smart and sensor systems, antennas and transmission, smart cards, energy efficient design and operation, sustainable development

Panagiotis Tolis is a graduating student of the program BSc (Hons) Computer Networks and Security of Informatics School of Mediterranean College in conjunction with Derby University. He is actively employed in computer networks and security, possessing CCNA. Research interests: Network protocols, network security, cryptosystems

Panagiotis Perakis received his BSc degree in Physics, his MSc degree in ICT and his PhD in Computer Science from the National and Kapodistrian University of Athens, Greece. After many years in the Information Technology business, as a software engineer, CTO and entrepreneur, he made a shift to his career for the academia. He has been actively involved in various EU funded research projects and worked as a research and teaching assistant at the Dept. of Informatics and Telecommunications of the National and Kapodistrian University of Athens (NKUA), and as a research associate at the Dept. of Computer Science of the Norwegian University of Science and Technology (NTNU). Currently, he serves as Head of the School of Computing at Mediterranean College, Athens, Greece. He has a strong publication record in top scientific journals and conference proceedings. He serves as a scientific journal reviewer, and as a research and innovation assessor of the Greek General Secretariat of Research and Technology (GGET). He is a member of the Hellenic Physical Society, since 1987, and of the IEEE Computer Society, since 2011. Research interests: computer graphics, computer vision, image processing, pattern recognition, artificial intelligence and physics-based modeling and simulation.