Joint Conference of 1996 International Computer Symposium
December 19~21, Kaohsiung, Taiwan, R.O.C.

A Smart Card Model Password Authentication Scheme Based on Lucas Functions

Tzong-Chen Wu and Guey-Fuh Sheu

Department of Information Management
National Taiwan Institute of Technology
Taipei, Taiwan 106, Republic of China

Abstract

The Lucas function is a special form of second-order linear recurrence relation using a large public integer as the modulus. With the usage of tamper-free smart cards, we present a remote login authentication scheme based on Lucas functions. Like the RSA system, the Lucas-based cryptosystem is susceptible to the chosen-message forgery attack. However such vulnerability does not affect the security of the proposed scheme for authentication purpose only. The proposed scheme has the following characteristics: (1) users can freely choose their passwords with fixed or variable length; (2) users can perform on-line registration for their own smart cards and passwords with the password center; (3) each user and his own smart card has unique one-to-one correspondence, hence no identity for the user is required; (4) the authentication server can easily verify a login request without verification tables or preserved secrets; (5) any adversary cannot successfully replay an intercepted legal login request in the later time; (6) any adversary cannot successfully forge a legal login request with an illegally possessed smart card.

Keywords: *password authentication, remote login, on-line registration, Lucas function, smart card.*

1. Introduction

The main tasks of the authentication server (*AS*) in a computer system are to identify the user who requests access and to ascertain that the login user is legal or not. In general, three approaches are usually adopted in computer systems for user authentication [6, 17, 26]:

(1) by the secret information only known to the users, such as passwords or facts;
(2) by a possessed special "token" which is hard to duplicate, such as mechanical keys or smart cards.
(3) by user's biometric characteristic, such as written signatures or fingerprints.

To enhance the secrecy and privacy, any effective user authentication mechanism should associate to at least two approaches stated above [26]. Identifying user's biometric characteristics requires extra hardware costs and is time-consuming. There is no doubt that authenticating passwords in smart card model is the most acceptable approach for user authentication, since its inexpensiveness, simple implementation, and ease of use.

In conventional password authentication scheme, each user registers his public identity (*ID*) and secret password (*PW*) with the *AS*, and the *AS* keeps a secret password file containing all registered users' *ID*s and *PW*s. When logging the system, the user submits his *ID* and *PW* to the *AS*. The *AS* verifies the login request by checking if the submitted *ID* and *PW* are identical to the corresponding entries of the password file. It goes without saying that directly storing the plain password file is insecure. Further, an intruder may intercept a legal login request and disclose the plain password, and hence successfully replays it in later time.

To overcome the drawbacks stated above, an alliterative approach is that the *AS* uses an available cryptographic technique, such as one-way function [21] or encryption algorithm [4], to encode users' passwords into test patterns (*TP*s), and stores *ID*s and *TP*s in a public directory, namely the verification table. When logging the system, the user uses the same cryptographic technique to encode his password into a test pattern and submits his *ID* and *TP* to the *AS*. The *AS* verifies the login request by checking if presented *ID* and *TP* are identical to the corresponding entries of the verification table. Most of the previously proposed password authentication schemes adopt such approach [2, 7, 8, 9, 13, 14, 18]. However, this approach still has the disadvantage that an intruder may successfully replay an intercepted legal login request in the later time. Besides, the *AS* requires extra memory space and management effort for maintaining the verification table.

Smart cards are some kinds of temper-free security devices, and are prevailingly used in computer and communication network applications [5, 22, 27]. A smart card contains a microprocessor, read-only memory (ROM), random access memory (RAM), nonvolatile memory (NVM), and input/output ports [22]. For security considerations, the smart card should operate under exclusive control of the software in the on-board ROM. Hence, the smart card has the capability to withstand unauthorized access to the CPU, the memory, the buses, and the data stored within the device. Chang and Wu [3] and Wu and Chang [28] proposed two password authentication schemes with the usage of smart cards. Both of these two schemes require no verification tables and are useful for remote login under insecure channels. In the Chang-Wu scheme, the users' passwords are generated by the system. The system-generated password preserves randomness to secrecy, however loses mnemonics to practical usage [16]. In the Wu-Chang scheme, some preserved secrets for the *AS* should be involved in the authentication stage, and hence may increase the risk of breaking the system.

The Lucas function is a special form of second-order linear recurrence relation using a large public integer as the modulus [21]. Recently, Smith and Lennon [23, 25] proposed a new public key cryptosystem, namely the LUC system, based on Lucas functions. Since then, the properties of Lucas functions in cryptographic applications have been studied extensively [1, 9, 11, 12, 24, 29]. The main selling point of the Lucas-based cryptosystems is that they are not formulated in terms of exponentiation [1]. It is believed that any successful attack, for instance, the chosen-message forgery [1], on the Lucas-based cryptosystems would lead to those based on the exponentiation [11, 12]. Although the security of the LUC system is not well convinced till now, the significance of the Lucas functions in cryptographic applications is unsusceptible.

With the usage of tamper-free smart cards, we intend to propose a remote login authentication scheme based on Lucas functions. The proposed scheme has the following characteristics:

(1) users can freely choose their passwords with fixed or variable length;
(2) users can perform on-line registration for their own smart cards and passwords with the password center;
(3) each user and his own smart card has unique one-to-one correspondence, hence no identity for the user is required;
(4) the authentication server can easily verify a login request without verification tables or preserved secrets;
(5) any adversary cannot successfully replay an intercepted legal login request in the later time;
(6) any adversary cannot successfully forge a legal login request with an illegally possessed smart card.

2. Preliminaries

In this section, we first introduce some useful properties of the Lucas function. Also, a known Lucas-based cryptosystem, the LUC system [22, 23], is described.

2.1. The Lucas function

Let G be a finite field and $m \in G$. The Lucas function is a special form of a second-order linear recurrence relation [22]:

$$V_n(m) = mV_{n-1}(m) - V_{n-2}(m), \text{ for } n \geq 2,$$

where $V_0(m) = 2$ and $V_1(m) = m$. The sequence $< V_n(m) >$ is called the Lucas sequence generated by m. The famous Fibonacci sequence is a variation of the Lucas sequence [21].

The Lucas function has several useful properties in cryptographic applications [9, 11, 12, 24]. Two of the significant properties of the Lucas function applied to the design of cryptographic applications are described as below:

Property 1: If we calculate the sequence $< V_n(m) >$ out of some point n and take the result modulo N (which is a composite of two primes), then we have the same result as if we had applied the modulo operation at each step in the sequence. That is,

$$V_n(m) \bmod N = V_n(m \bmod N).$$

Property 2: The calculation of a composite Lucas sequence produced by two Lucas sequences preserves the communicative relationship. That is,

$$V_{ab}(m) = V_a(V_b(m)) = V_b(V_a(m))(\bmod N).$$

The reader is encouraged to learn more useful properties about the Lucas function in the literature [1, 9, 11, 12, 23, 24, 25].

2.2. The LUC system

The overall structure of the LUC system is similar to the well-known RSA system [19]. The encryption and decryption of a message is achieved using the Lucas sequences generated by the message. Differentiating from the RSA system, the decryption key for the Lucas-based cryptosystem may either be message-dependent or message-independent [1]. Using message-dependent decryption key may make the generation of signatures more efficient [1]. The LUC system with message-dependent decryption key is described in the following.

Suppose user A wants to send a secret message to user B. First of all, B chooses two large primes p and q, computes $N = pq$, and selects an integer $e \in Z_{\varphi(N)}$ such that GCD$((p^2-1)(q^2-1), e)$ = 1, where φ is Euler's totient function [21] and GCD means the greatest common divisor. Afterwards, B publishes (e, N) as his public key, while keeps p and q secret. A encrypts the message m, for $m < N$ and GCD(m, N) = 1, by computing the ciphertext $c = V_e(m) \bmod N$. Here, c is the e-th point of the Lucas sequence generated by m. When received the ciphertext c, B performs the following steps to recover m:

Step 1. Compute $D = c^2 - 4$.

Step 2. Compute $S(N)$ = LCM(p-$L(D, p)$, q-$L(D, p)$), where LCM means the least common multiplier and $L(a, b)$ is the Legendre symbol defined as

$$L(a,b) = \begin{cases} 0 & \text{if } b \text{ divides } a \\ 1 & \text{if } a \text{ is a quardratic residue mod } b \\ -1 & \text{if } a \text{ is not a quardratic residue mod } b \end{cases}$$

Step 3. Compute $d = e^{-1} \bmod S(N)$, where e^{-1} is the inverse of e modulo $S(N)$.

Step 4. Recover the message as $m = V_d(c) \bmod N$.

References 23 and 25 give more details of the proof that $V_d(V_e(m)) = V_e(V_d(m)) = m \pmod N$ for $ed \equiv 1 \bmod S(N)$. Note that d is a function of m. For a given e, there are only four possible values of $S(N)$:

LCM(p - 1, q - 1), for $L(D, p) = 1$ and $L(D, p) = 1$,
LCM(p- 1, q + 1), for $L(D, p) = 1$ and $L(D, p) = -1$,
LCM(p + 1, q - 1), for $L(D, p) = -1$ and $L(D, p) = 1$,
LCM(p + 1, q + 1), for $L(D, p) = -1$ and $L(D, p) = -1$.

Therefore, there are only four possible solutions for d satisfying $ed \equiv 1 \bmod S(N)$:

$d_1 = e^{-1} \bmod \text{LCM}(p - 1, q - 1)$,
$d_2 = e^{-1} \bmod \text{LCM}(p - 1, q + 1)$,
$d_3 = e^{-1} \bmod \text{LCM}(p + 1, q - 1)$,
$d_4 = e^{-1} \bmod \text{LCM}(p + 1, q + 1)$.

and these d_i's can be calculated at the time when e is chosen. For the decryption of a ciphertext as $c = V_e(m) \bmod N$, the receiver should first determine the values of $L(c^2-4, p)$ and $L(c^2-4, p)$ for choosing the proper d among $\{d_1, d_2, d_3, d_4\}$, and then recovers the message as $m = V_d(c) \bmod N$ without adding significant computational load.

The LUC system with message-independent decryption key can be simply presented in terms of an RSA-like system. Each user publishes the produce N of two large primes p and q, and an encryption key e with $\mathrm{GCD}(e, (p^2-1)(q^2-1)) = 1$. The corresponding decryption key d such that $ed \equiv 1 \bmod (p^2-1)(q^2-1)$ is kept secret. The ciphertext of a message m is computed as $c = V_e(m) \bmod N$. To decrypt the message, the user computes $m = V_d(c) \bmod N$. Like the RSA system, the signature for m is computed as $s = V_d(m) \bmod N$, which can be verified by checking that $m = V_e(s) \bmod N$. Note that the signature scheme for the LUC system takes more computation than the RSA system does. Yen and Laih [29] proposed an efficient algorithm for the signature scheme for the LUC system. Besides, a parallel architecture for Yen and Laih's algorithm is developed in the same paper.

It was pointed out [1, 9] that, like the RSA system, the original LUC system is susceptible to the chosen-message forgery attack. Bleichenbacher et. al. [1] demonstrated an effective chosen-message forgery attack on the LUC signature scheme. However such vulnerability does not affect the security of the proposed scheme for authentication purpose only, as discussed in Section 4.

3. The Proposed Scheme

Throughout the paper, we adopt the presentation of message-independent LUC system for describing the proposed scheme. The proposed scheme is divided into four stages: the system initialization stage, the smart card registration stage, the user login stage and the server authentication stage. Four primary roles are involved in these four stages: the password center (*PC*), the user, the smart card and the authentication server (*AS*). In the system initialization stage, the *PC* generates necessary public and secret parameters for the system and the smart card. Initially, the *PC* assigns an initial password for each smart card, and the smart card with its initial password will be delivered to the user by hand before registration stage. Other parameters for the smart card are burned onto its ROM to let only the smart card itself can read these parameters [27]. In the user registration stage, the user performs on-line registration for his own smart card (with its initial password) and a chosen password with the *PC*. The *PC* produces a test pattern, which is corresponding to the chosen password for the registered user, and stores the test pattern in the NVM of the smart card. All successful on-line registrations for the smart cards are recorded in a registration status file maintained by the *PC*. In the user login stage, the logging user first attaches his smart card to a terminal which connects to the *AS* and inputs his identity and password. After that, the smart card generates a remote login request and transmits it to the *AS*. In the server authentication stage, the *AS* verifies the login request without the assistance of verification tables or preserved secrets. We describe these four stages in details as follows:

[System initialization stage]

The *PC* performs the following tasks for system setup:

Step 1. Define N_s, e_s and d_s for the system.

Step 2. Define N_c, e_c and d_c for the smart cards, where $N_c < N_s$.

Step 3. Choose an available one-way hashing function h, such as MD5 [17].

Step 4. For each smart card, do the following:

(4-1). Assign an identity ID_c and an initial password PW_c for the smart card.

(4-2). Burn ID_c, (N_s, e_s), (N_c, e_c), h, and d_c onto the ROM of the smart card.

(4-3). Compute a test pattern for PW_c as
$$A_c = V_{d_s}(V_{e_c}(h^2(PW_c)) \oplus h(ID_c) \bmod N_c) \bmod N_s$$
and put it in the NVM of the smart card, where $\oplus$ denotes the exclusive-or operator.

Step 5. Publish N_s, e_s, N_c and h; while keep d_s secret.

Note that all smart cards have the same N_s, e_s, N_c and h; while they may have different ID_c, A_c, e_c and d_c. The secret parameter d_s is used for generating an authentic test pattern for the owner of the smart card in the user registration stage. Any user who wants to join the system should first possess a legal smart card with its initial password PW_c delivered by the *PC*. It should be noticed that the initial password for the smart card is used only once for registration.

[Smart card registration stage]

Suppose a new user U_i wants to join the system. First of all, U_i attaches his own smart card to a terminal that connects to the *PC*. Afterwards, U_i informs the smart card for invoking an on-line registration for his own smart card and the chosen password PW_i. The on-line registration is performed as follows:

Step 1. The smart card requests U_i to key in the initial password PW_c for the smart card and the chosen password PW_i for later login.

Step 2. The smart card requests a timestamp T from the *PC*.

Step 3. The smart card computes
$$Y_c = V_{d_c}(h(PW_c) \oplus h(T)) \bmod N_c, \text{ and}$$
$$Y_i = V_{e_s}(V_{d_c}(h(PW_i) \oplus h(PW_c) \oplus h(T)) \bmod N_c) \bmod N_s.$$

Step 4. The smart card sends $\{ID_c, Y_c, Y_i, A_c, e_c, T\}$ to the *PC*.

Step 5. The *PC* validate the expiration of the timestamp T by checking that $|T' - T| \le \delta$, where T' is the arrived time for the request and δ is the acceptable transmission delay between the *PC* and the terminal.
If the transmission delay is not acceptable, then reject the registration request.

Step 6. The *PC* looks up the registration status file and checks if ID_c has been successfully registered by some user.
If it does, then the *PC* terminates the registration stage.

Step 7. The *PC* verifies Y_c and A_c by checking that
$$V_{e_s}(A_c) \bmod N_s = V_{e_c}(h(V_{e_c}(Y_c) \oplus h(T))) \oplus h(ID_c) \pmod{N_c}.$$
If the equivalence does not hold, then terminate the registration stage.

Step 8. The *PC* computes
$$Z_i = V_{e_c}(V_{d_s}(Y_i) \bmod N_s) \oplus V_{e_c}(Y_c) \bmod N_c,$$
and
$$A_i = V_{d_s}(V_{e_c}(Z_i) \oplus h(ID_c) \bmod N_c) \bmod N_s,$$
and replies $\{ID_c,\ A_i\}$ to the smart card.

Step 9. The smart card verifies A_i by checking that
$$V_{e_s}(A_i) \bmod N_s = V_{e_c}(h(PW_i)) \oplus h(ID_c) \pmod{N_c}.$$
If the equivalence does not hold, then the smart card terminates the registration stage; otherwise replaces A_c with A_i in the smart card and informs the *PC* to record necessary registration information for the smart card in the registration status file.

Note that validating the expiration of the timestamp *T* is to avoid the attack that an adversary may replay an intercepted legal registration request to the *PC* for successfully impersonating some registering smart card.

Having successfully finished the on-line registration stage, the registered user and his own smart card have one-to-one correspondence. The authentic message A_i in the smart card is regarded as the test pattern for ID_c and PW_i. Hence, a legal remote login request can only be produced by the user and his own smart card.

It is to see that the registering challenges and responses communicated between the smart card and the *PC* reveal no plain passwords (the initial password for the smart card and the chosen password for the user), even for the *PC* only knows their ciphertext forms. The reason for maintaining a registration status file by the *PC* is to prevent the adversary with an illegally possessed smart card from successfully plotting an on-line registration again. Hence, every user can register his own smart card with its initial password only once.

In the smart card registration stage, the check for the authentic message A_c in Step 7 is to let the *PC* ensure that the registration is invoked by a legal smart card, and the check for the authentic message A_i in Step 9 is to let the smart card ensure that the chosen password PW_i for U_i is successfully registered. The equivalence for the check in Step 7 of the smart card registration stage can be directly derived from Step (4-3) of the system initialization stage.

In the following, we show the equivalence for the check in Step 9 of the smart card registration stage. First of all, from Step 3, we have
$$V_{e_c}(Y_c) = h(PW_c) \oplus h(T) \pmod{N_c}, \text{ and}$$
$$V_{d_s}(Y_i) \bmod N_s = V_{d_c}(h(PW_i) \oplus h(T)) \pmod{N_c}.$$

Thus, from Step 8, we have
$$Z_i = h(PW_i) \bmod N_c,$$
which implies
$$A_i = V_{d_s}(V_{e_c}(Z_i) \oplus h(ID_c) \bmod N_c) \bmod N_s$$
$$= V_{d_s}(V_{e_c}(h(PW_i)) \oplus h(ID_c) \bmod N_c) \bmod N_s.$$
Consequently, the equivalence for the check in Step 9 can be conducted straight away.

[User login stage]

When logging the system, U_i first attaches his smart card to a terminal that connects to the *AS*, and keys in PW_i. Afterwards, the smart card performs the following tasks:

Step 1. Request a timestamp *T* from the *AS*.

Step 2. Compute a test pattern with respect to PW_i and *T* as
$$C_i = V_{d_c}(h(PW_i) \oplus h(T)) \bmod N_c.$$

Step 3. Construct a remote login request $\{ID_c,\ A_i,\ C_i,\ e_c,\ T\}$ for U_i and transmit it to the *AS*.

[Server authentication stage]

Having received $\{ID_c,\ A_i,\ C_i,\ e_c,\ T\}$ sent from the user, the *AS* performs the following tasks to verify the login request:

Step 1. Validate the expiration of the timestamp *T* by checking that $|T'' - T| \le \sigma$, where T'' is the arrived time for the request and σ is the acceptable transmission delay between the *AS* and the terminal.
If the transmission delay is not acceptable, then reject the login request.

Step 2. Validate the login request by checking that
$$V_{e_s}(A_i) \bmod N_s = V_{e_c}(V_{e_c}(C_i) \oplus h(T)) \oplus h(ID_c) \pmod{N_c}.$$
If the equivalence holds, then accept the login request; otherwise reject the login request.

Note that the *AS* verifies the remote login request without the assistance of verification tables or preserved secrets. Since the user and the smart card have a unique one-to-one correspondence. Anyone observing the login request cannot have the knowledge of identity for the logging user, if the observer does not know the corresponding owner of the smart card with identity ID_c. The equivalence for the check of Step 2 in the server authentication stage is conducted as follows: From Step 9 in the smart card registration stage, we have
$$V_{e_s}(A_i) \bmod N_s = V_{e_c}(h(PW_i) \oplus h(ID_c) \pmod{N_c}.$$
Further, from Steps 2 and 3 in the login stage, we have
$$V_{e_c}(V_{e_c}(C_i) \oplus h(T)) \oplus h(ID_c) \bmod N_c$$
$$= V_{e_c}(B_i \oplus h(T)) \oplus h(ID_c) \bmod N_c$$
$$= V_{e_c}(h(PW_i) \oplus h(T) \oplus h(T)) \oplus h(ID_c) \bmod N_c$$
$$= V_{e_c}(h(PW_i)) \oplus h(ID_c) \bmod N_c$$
$$= (V_{e_s}(A_i) \bmod N_s) \bmod N_c.$$
Consequently, the equivalence for the check in Step 2 of the server authentication stage can be conducted straight away.

4. Analysis

4.1. Crypto Analysis

We analyze some possible attacks plotted by the adversary with the knowledge about the legal registering messages for the user, the legal remote login requests, and the other public parameters. These possible security issues are discussed as below.

Issue 1: Reveal secret parameters for the system and the smart card.

An adversary may try to reveal the secret parameter d_s for the PC from N_s and e_s. However, the adversary should first know p_s and q_s, which is based on the difficulty of factoring N_s, for computing d_s [22]. Similarly, the adversary will face the same difficulty for revealing the secret parameter d_c for the smart card.

Issue 2: Reveal passwords from intercepted messages or login requests.

In both the smart card registration and user login stages, plain passwords, i.e., PW_c and PW_i, are protected by the one-way function h, and are represented in the ciphertext forms $h(PW_c)$ and $h(PW_i)$. Therefore, any adversary, even the PC or the AS, reveals no useful knowledge about these plain passwords from intercepted registering messages or login requests.

Issue 3: Plot re-registration attack with illegally possessed smart card.

Unless knowing the initial password PW_c that is generated by the PC in the system initialization stage, anyone cannot register the smart card at the first time. Once a smart card has been registered with respect to certain user U_i, the content of A_i has one-to-one correspondence to both ID_c and PW_i with respect to the registering timestamp T. Further, the registration status for the smart card and its holder will be recorded in the registration status file maintained by the PC. Thus, the checks in Steps 5 and 6 in the smart card registration stage will preclude the adversary that makes an attempt on re-registration with an illegally possessed smart card (without knowing PW_c).

Issue 4: Replay a legal login request without the assistance of a smart card.

An adversary may intercept a legal login request $\{ID_c, A_i, C_i, e_c, T\}$ and replay it to the AS for impersonating U_i in the later time. However, such attack will be precluded by the check of Step 1 of the server authentication stage.

Issue 5: Forge a legal login request without smart card.

Clearly, any adversary knows the secret key d_s for the PC or the secret key d_c for the smart card, then he could freely forge either A_i or C_i for constructing a legal login request at any time. The security of d_s and d_c is covered by Issue 1.

Issue 6: Forge login request with an illegally possessed smart card.

Unless the owner of the smart card presents the right password for himself, the test pattern C_i calculated by the smart card in Step 2 of the user login stage cannot pass the check in Step 2 of the server authentication stage. Suppose the adversary does not know d_s, d_c and PW_i in advance, and tries to forge a legal login request with the assistance of an illegally possessed smart card. By the discussion of Issue 2, the adversary may obtain $h(PW_i)$ from an intercepted login request. It is to see that, for instance, if the adversary chooses an acceptable timestamp T^* and obtains the signature for $h(PW_i) \oplus h(T^*)$ with the key d_c, then he can easily construct a legal login request $\{ID_c, A_i, C_i^*, T^*, e_c\}$ that can pass the checks in the server authentication stage. Such attack is called the chosen-message forgery attack. In order to successfully plot the chosen-message forgery attack state above, the adversary should submit the corresponding $h(PW_i)$ and T^* to the smart card for signing $h(PW_i) \oplus h(T^*)$ with the key d_c. The security of revealing passwords from intercepted registering messages or login requests are discussed in Issue 2. Thus, the chosen-message forgery attack cannot work without knowing PW_i in advance.

4.2. Computational Analysis

Let t_{LUC} be the time for calculating the Lucas function, t_h be the execution time for the one-way hashing function h. The computational complexity for the user login stage and the server authentication stage is analyzed as follows:

The computational complexity for the user login stage is $t_{LUC} + 2\times t_h$ The computational complexity for the server authentication stage is $3\times t_{LUC} + 2\times t_h$. By the precomputation of $h(PW_i) \bmod N_c$, the computation complexity for the user login stage can be further reduced to $t_{LUC} + t_h$, hence the proposed scheme is suitable to the application with the usage of smart cards.

5. Conclusions

With the usage of the LUC system and tamper-free smart cards, we have presented a new password authentication scheme for verifying remote login requests. Different from the previously proposed schemes with the same approach in smart card model, such as that in [3] and [28], this new proposed scheme has the following features:

(1) Each user and his own smart card have unique one-to-one correspondence, hence no identity for the user is required.
(2) The proposed scheme provides on-line registration for both the users and the smart cards; whereas most of the previously proposed schemes only provide off-registration for the users.
(3) The authentication server can easily verify remote login requests without the assistance of verification tables or preserved secrets.

Acknowledgment

Partial of this work is supported by the National Science Council, Taiwan, Republic of China, under Contract number NSC85-2213-E011-053.

References

[1] Bleichenbacher, D., Bosma, W., and Lenstra, A.K.: "Some remarks on Lucas-based cryptosystems", *Advances in Cryptology - CRYPTO'95*, Springer-Verlag, Berlin, 1995, pp. 386-396.

[2] Chang, C.C. and Huang, S.J.: "Cryptographic authentication of passwords", *Proceedings 1991 IEEE International Carnahan Conference on Security Technology*, Taipei, Taiwan, R.O.C., 1991, pp. 126-130.

[3] Chang, C.C. and Wu, T.C.: "Remote password authentication with smart cards", *IEE Proceedings-E*, Vol. 138, No. 3, 1991, pp. 165-168.

[4] *Data Encryption Standard*, Federal Information Processing Standard (FIPS), Publication 45, January 1977, National Bureau of Standards, U.S. Department Commerce.

[5] Davies, D.W.: "Smart cards, digital signatures, and negotiable documents", *Processings of the International Conference on Secure Communications Systems*, London, UK, 1984, pp. 1-4.

[6] Davies, D.W. and Price, W.L.: *Security for Computer Networks - An Introduction to Data Security in Teleprocessing and Electronic Funds Transfer*, Second Edition, John Wiley & Sons, Chichester, 1989.

[7] Evans, Jr., A., Kantrowitz, W. and Weiss, E.: "A user authentication scheme not requiring secrecy in the computer", *Communications of the ACM*, Vol. 17, No. 8, 1974, pp. 437-442.

[8] Feistel, H., Notz, W.A. and Smith, J.L.: "Some cryptographic techniques for machine to machine data communications", *Proceedings of IEEE*, Vol. 63, No. 11, 1975, pp. 1545-1554.

[9] Horster, P., Petersen, H., Michels, M.: "Digital signature schemes based on Lucas functions", University of Technology Chemnitz-Zwickau, Technical Report TR-95-1; also appears in: *Communications and Multimedia Security, IT-Sicherheit'95, Joint Working Conference IFIP TC-6 TR-11 and Austrian Computer Society*, Graz, September 1995, pp. 1-13.

[10] Hwang, T.Y.: "Passwords authentication using public key encryption", *Proceedings 1983 IEEE International Carnahan Conference on Security Technology*, Zurich, Switzerland, 1983, pp. 35-38.

[11] Laih, C.S., Tu, F.K., and Tai, W.C.: "Remarks on the LUC public key system", *Electronics Letters*, Vol. 30, No. 2, 1994, pp. 123-124.

[12] Laih, C.S., Tu, F.K., and Tai, W.C.: "On the security of the Lucas function", *Information Processing Letters*, Vol. 53, 1995, pp. 243-247.

[13] Lamport, L.: "Password authentication with insecure communication", *Communications of the ACM*, Vol. 24, No. 11, 1981, pp. 770-772.

[14] Lennon, R.E., Matyas, S.M., and Meyer, C.H.: "Cryptographic authentication of time-invariant quantities", *IEEE Trans, on Communications*, Vol. COM-29, No. 6, 1981, pp. 773-777.

[15] Lin, C.H., Chang, C.C, Wu, T.C., and Lee, R.C.T.: "Password authentication using Newton's interpolating polynomials", *Information Systems*, Vol. 16, No. 1, 1991, pp. 97-102.

[16] Menkus, B.: "Understanding the use of passwords", *Computers and Security*, Vol. 7, No. 2, 1988, pp. 132-136.

[17] Pfleeger, C.P.: *Security in Computing*, Prentice-Hall, Englewood Cliffs, New Jersey, 1989.

[18] Purdy, G.P.: "A high security log-in procedure", *Communications of the ACM*, Vol. 17, No. 8, 1974, pp. 442-445.

[19] Rivest, R.: "The MD5 message digest algorithm", RFC 1321, April 1992.

[20] Rivest, R., Shamir, A., and Addleman, L.: "A method for obtaining digital signatures and public key cryptosystems", *Communications of the ACM*, Vol. 21, No. 2, 1978, pp. 120-128.

[21] Schroeder, M. R.: *Number Theory in Science and Communication*, Second Edition, Springer-Verlag, Berlin, 1990.

[22] Simmons, G.J.: *Contemporary Cryptology*, the Institute of Electrical and Electronics Engineers, Inc., New York, 1992.

[23] Smith, P.: "LUC public key encryption: A secure alternative to RSA", *Dr. Dobb's Journal*, January 1993, pp. 44-49.

[24] Smith, P.: "Cryptography without exponentiation", *Dr. Dobb's Journal*, April 1994, pp. 26-30.

[25] Smith, P. and Lennon, M.: "LUC: A new public key system", *Proceedings of the 9th IFIP Symposium, IFIP/Sec '93, Toronto*, May 1993, pp. 97-110

[26] Stallings, W.: *Network and Internetwork Security - Principles and Practice*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey. 1995.

[27] Sternglass, D. :"The future is in the PC cards", *IEEE Spectrum*, Vol. 29, No. 6, 1992, pp. 46-50.

[28] Wu, T.C. and Chang, C.C.: "A password authentication scheme based on discrete logarithms", *International Journal of Computer Mathematics*, Vol. 14, No. 3+4, 1991, pp. 31-38.

[29] Yen, S.M., Laih, C.S.: "Fast algorithm for the LUC digital signature computation", *IEE Proceedings-E.*, Vol. 142, No. 2, 1995, pp. 165-169.