Integration of user authentication and access control

L. Harn
H.-Y. Lin

Indexing terms: User authentication, Access control

Abstract: User authentication and access control are both necessary mechanisms for data protection in a computer system. Traditionally, they are implemented in different modules. In this paper, a new solution is presented to provide both user authentication and access control in a single module to avoid any possible security breach between these two protection mechanisms. The secret information required for the whole system is minimised and the difficulty of password comprising increased to improve system security. More importantly, with time complexity of implementation almost equivalent to that found in the normal public key based password authentication schemes and limited extra storage space, both user authentication and access control can be achieved at the same time.

1 Introduction

Computer and electronic technologies have developed together to encourage a dramatic increase in the volume and speed of information processing and distribution. As a result, a vast amount of computer resources, i.e. personal computers, terminals, printers, databases, host systems, and data stored in electronic devices, etc., are now combined through networks to provide varieties of services. Organisations in both the public and private sectors are becoming increasingly dependent on these resources. Without appropriate protection, these resources are susceptible to unauthorised access.

Traditionally, in order for a system to make a decision about whether an outside user has access privilege to the resources in the system, the outside user must first submit a secret password to the system for password verification. If the user passes the verification, he/she is admitted into the system and will be assigned a unique identification. This process is called user authentication. Because password verification is the most popular technique for user authentication, our discussion on user authentication will refer to password verification for the rest of this paper. Once the unique user identification is determined by the system, the secret password will not be referred to again and this unique user identification will be used repeatedly to make access decision by checking against his/her designated access rights stored in the system. This validation of access privilege associated with each unique identification is called access control.

Obviously, any security breach in either the user authentication or access control may threaten the security of the system. We also know that the security of access control relies not only on itself but also on the correct identification which is determined by user authentication mechanism. So, why do we not implement these two mechanisms in a single module instead of in two separate ones? From the security aspect, it would be a better choice. Today, many security problems result from the penetration of, or defects in, one or other of these two mechanisms. However, most of them come only from the failure of a single mechanism, i.e. the crackdown of some passwords, the backdoors in the system, or carelessness of the users, etc., which all fall into these two categories. By integrating these two mechanisms in a single module, the attacks on each mechanism as mentioned above will not occur again.

Paper 8573E (C5), first received 15th October 1990 and in revised form 7th October 1991

The authors are with the Computer Science Telecommunications Program, University of Missouri — Kansas City, Kansas City, MO64110, USA

2 Review of password authentication scheme

Password authentication is one of the most common and elementary operations used to protect resources from users' unauthorised access. In the authentication process, every user submits his/her public identification (*ID*) and the secret password, while logging into the system. By using this public *ID* as a key, the system fetches the stored password to determine whether or not it matches with the submitted one. If it does, the user is granted the right to use the system. Otherwise, he/she is rejected.

The traditional way to achieve password authentication is done by storing a password table in the host system for login verification. Since each user's password is stored in the table directly, disclosure of this table will destroy the system security completely.

To avoid the above weakness, new versions of operating systems, like UNIX, store passwords in ciphertext form by using a one-way function (i.e. DES). In this way, passwords are free of unauthorised exposure even if their corresponding cipher text is revealed to the public [11]. This is because it is impossible to invert a one-way function to derive the password.

Unfortunately, the above approach is not the ultimate solution to the password authentication problem. As we know, most users like to choose short and easily remembered passwords instead of long and obscure ones. So, it is not difficult for hackers to build a dictionary by collecting all possible passwords which are derived from the user's personal information (i.e. user's name, address, birthday, telephone number, etc), and some frequent English words. After encrypting these possible passwords, one can compare the resulting ciphertext with those stored in the encrypted password file. This approach to crack the password has been widely used. The most successful one is found in the Internet worm which intruded approximately 6000 computers nationwide in November, 1988 [14].

Generally speaking, security of the password authentication scheme relies on the combination of three factors: the secrecy and integrity of passwords or encrypted passwords, the secrecy of encryption keys for generating passwords, and the time complexity of the encryption algorithm. Since most systems allow users to choose their own passwords, and every user has the tendency to choose a short and easily remembered password, it becomes very difficult for the system to protect the secrecy of passwords from the attack of clever hackers. On the other hand, many computer systems use the DES scheme to encrypt the password file, but the efficiency of the DES scheme may also lend itself to the possibility of compromising the passwords. Most recently, Biham and Shamir [15, 16] introduced the notion of differential cryptanalysis based on chosen plaintext attacks to break some cryptosystems similar to DES. Therefore, the security of DES is still in doubt.

In 1976, Diffie and Hellman [4] proposed the revolutionary public key concept which has significant advantages over the traditional encryption schemes. Since then, several public key schemes have been introduced [5, 10, 12] and some of them have been used to developed password authentication schemes [7, 9]. There are two common features among these schemes; each user's password is generated from the system and the size of the password is quite long (i.e. at least 512 bits). Secrecy and integrity of the password file in these systems is now enhanced and it also eliminates the possible attack from the clever hackers.

In this paper, we propose a similar cryptographic approach which uses a one-way trapdoor cryptographic function to bind the user secret password and user identification together. Thus we remove the necessity to store an encrypted password file in the system. This approach has greatly reduced the risk of cracking the password by attacking the encrypted password file. In addition, we reduce the amount of secret information stored in the host system.

3 Review of access control

Access control is also a protection mechanism which has been applied to subjects (accessor) and objects (information resource) in the system. It ensures that all accesses to objects are authorised by regulating different privileged operations. The access matrix model [3] has been used widely to decribe the protection mechanism. With each row corresponding to a subject and each column to an object, the privileges to access information resources by each accessor are well arranged. Subjects are accessors in the system, like users, processors, or programs, etc. Objects are resources in the system, like files, user account, or segment of memory, etc.

Access matrix model can also reflect the modifications of access privileges which the subjects possess on the objects. The modifications include the insertion and deletion of subjects and objects, and the granting and revoking of access privileges which may include read, write, execute, open, etc. If one subject has no access right to an object, the corresponding entity in the matrix is assigned a 'null' value. Most of the entities in the matrix are nulls because each subject is usually allowed only to access a subset of the objects. Therefore, to implement the access control mechanism by storing the whole access matrix results in very low utilisation of storage. On the other hand, the memory needed for the whole matrix becomes impractically large when the number of subjects and objects is sufficiently large.

In general, there are three approaches to implementing the access matrix. They are as follows.

(i) *The access list system* [13]. Each object, O, is assigned a list, L_O, with (S: subject name, r: access right) pairs. For each subject, S, which has access privilege, r, on the object, O, (S, r) is inserted into L_O. In this way, none of the entities with null values in the matrix will be stored, therefore, saving much storage space. This approach can also handle the revocation of subjects' access rights on an object simply by deleting the corresponding entities in the list. However, it suffers from the disadvantage of longer searching time on access lists since accesses of a subject to objects involve the searching of all of the access lists associated with these objects.

(ii) *The capability list system.* Each subject, S, is assigned a list, L_s, with (O: object name, r: access right) pairs. For each object, O, which can be accessed by subject, S, with privilege r, (O, r) is inserted into L_s. This approach does not have the same problem of long searching time as in the access list approach, but it suffers from disadvantages such as propagation, revocation and review problems [13].

(iii) *The key–lock matching system* [1, 6, 17]. By assigning a key to each subject, and a lock to each object, we can define a function which outputs the access privilege when a pair of (key, lock) is input upon each access request. This approach seems very efficient since it stores only locks/keys associated with objects/subjects. Unfortunately, in all algorithms proposed so far, the size of locks/keys is still larger than the access matrix and more efforts are needed to make this approach practical.

However, these methods have a common feature: one that leads to some of their main disadvantages in that they all require a memory list with a variable number of entries.

As far as we are aware, no one has ever proposed the cryptographic implementation of the access control mechanism. However, in this paper, we propose a new cryptographic approach which can achieve user authentication and access control simultaneously. By doing so, we enhance the integrity of the protection mechanism significantly.

4 Proposed cryptographic method

Inspired by the idea of the key generation scheme for the multilevel data security proposed by the same authors [8] and by Chick/Tavares's flexible access control with master key idea [2], we present a cryptographic scheme which can provide both user authentication and access control simultaneously, and the time complexity of this scheme is almost equivalent to those of public key based password authentication schemes [7, 9]. That is, access control can be achieved without extra computational overhead.

For the convenience of discussion, we assume that users and files are the only subjects and objects in the system, respectively, and users can access files with different privileges. These access privileges can be write, read, execute and open, etc. They are assumed to have a totally ordered relation. That is, the right to read the file implies the right to execute the file, the right to write the file implies the right to read and to execute the file, and so on. We assume that there are n users and m files in the system and the access privileges are represented by integers starting from 1. The lowest privilege is represented by 1 and the highest privilege is represented by r_{max}. Each user with identification ID_i is given a user number, UN_i, $i = 1, 2, \ldots, n$, and each file F_j is given a file number, $UF_j, j = 1, 2, \ldots, m$. The function of UN_is and UF_js will be explained later in the algorithm.

4.1 Algorithm

This new scheme can be divided into three phases: the registration phase, the verification phase, and the operational phase. In the registration phase, the system assigns a token to each file's access privilege, and a password to each user according to his/her access privileges over files, and keeps a secret master key for itself. In the verification phase, every access requested by a user is verified by examining whether the derived tokens for this access privilege from two different sources, the system master key and the submitted user password, are identical. In the operational phase, it provides the necessary functions to support modifications of the access matrix, i.e. insertion of files and users, deletion of files and users, granting of access privileges, and revoking of access privileges. More detailed procedures in these three phases are given below.

4.2 Registration phase

Each user needs to register and obtain a password from the host system. The system needs to check the user's ID and generate a corresponding password according to the user's access privileges. The procedures are described as follows:

Step 1: The system chooses two large secret primes p and q, which define the publicly known parameter $N = p \cdot q$, and $\alpha \in [2, N - 1]$, such that α and N are relatively prime i.e. $gcd(\alpha, N) = 1$.

Step 2: Assign to each file F_j an odd prime ef_j, $j = 1, 2, \ldots, m$, (i.e. we can start from the smallest value available) and compute the corresponding secret df_j with $ef_j \cdot df_j \bmod \phi(N) = 1$. By doing this, the token associated with privilege r of file F_j can be computed as

$$KF_{j,r} = \alpha^{df_j^r} \bmod N \quad j \in [1, m]$$

The master key for the system can be computed as

$$K_{\text{master}} = (\alpha)^{\Pi\, df_j^{r\,max}} \bmod N \quad \text{for } j = 1, 2, \ldots, m$$

The system keeps K_{master} secret and makes T publicly known, where

$$T = \Pi\, ef_j \quad \text{for } j = 1, 2, \ldots, m$$

Step 3: Assign to each user with ID_i, an odd prime, eu_i, $i = 1, 2, \ldots, n$, and calculate the corresponding secret du_i with $du_i * eu_i \bmod \phi(N) = 1$. Compute the password for each user ID_i, who has the total access privileges $r_{i,j}$ to file F_j, where $j = 1, 2, \ldots, m$, as

$$PW_i = \alpha^{du_i * \Pi\, df_j^{r_{i,j}}} \bmod N$$

Assign ID_i the secret password PW_i and compute the public information t_i as $t_i = \Pi ef_j^{r_{i,j}}$.

At the end of the registration phase, the system keeps p, q, K_{master} secret and T public, and each user keeps PW_i secret and t_i public. The values of eu_is and ef_js need not to be stored because they can be derived from UN_is and UF_js by some predefined function F.

4.3 Verification phase

In this paper, we have assumed that:

(i) it is not necessary that all accessors are properly identified prior to making any request (the access control mechanism will provide user identification as part of its operation)

(ii) accesses to files must go through the file system

(iii) data residing in main memory cannot be improperly read

(iv) the system's master key is stored in the monitor which cannot be modified by any accessor. The organisation of this new cryptographic system can be depicted in Fig. 1.

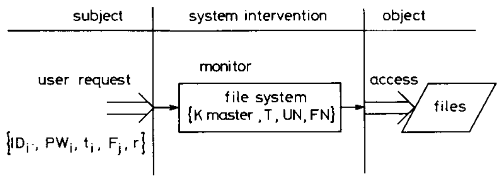

Fig. 1 *Organisation of proposed system*

When a user with ID_i, makes a request to access file F_j with privilege r, he/she submits the five-part information $(ID_i, PW_i, t_i, F_j, r)$, to the system. The system will first check if ID_i is a legitimate user, and then determines whether t_i is divisible by ef_j^r or not. If it is not, access request is denied. This is due to the fact that, in Step 3 of the registration phase, all the user privileges have been embedded in t_i. If t_i is divisible by ef_j^r, the system will further calculate the two tokens associated with privilege r from two different sources: one from the system's master key, K_{master}, and the other from the submitted user password, PW_i, to see whether these two values are identical. Suppose that the tokens derived from K_{master} and PW_i are V and V', respectively, then

$$\begin{aligned} V &= (K_{\text{master}})^{T^{r_{max}}/ef_j^r} \bmod N \\ &= (\alpha)^{(\Pi\, df_k^{r\,max}) \bullet (\Pi ef^{r\,max})/(ef_j^r)} \\ &= (\alpha)^{1/ef_j^r} \\ &= (\alpha)^{df_j} \\ &= KF_{j,r} \end{aligned}$$

and

$$V' = (PW_i)^{eu_i * t_i/ef_j^r} \bmod N$$

If PW_i and t_i are the valid ones associated with ID_i, then

$$\begin{aligned} V' &= (\alpha)^{(du_i * \Pi\, df_k^{r_{i,k}})(eu_i * \Pi ef_k^{r_{i,k}})/ef_j^r} \bmod N \\ &= (\alpha)^{1/ef_j^r} \bmod N \\ &= KF_{j,r} \bmod N \\ &= V \end{aligned}$$

If $V = V'$, the user is authenticated, and the access request is then authorised. Otherwise, the request is denied. These operations are depicted in Fig. 2.

4.4 Operational phase

In this phase, the system can provide some operations to support the modifications of the access matrix. Thus the scheme we present in this paper is a dynamic access control mechanism. Since all operations in this phase can be implemented simply by changing passwords of the related users, these operations will only be described as follows:

(i) *Insertion of a new user.* It can be implemented by just generating a new password according to the user's privileges.

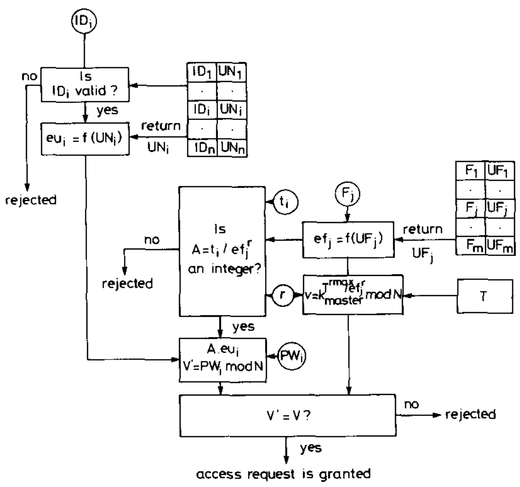

Fig. 2 *Integrated user authentication and access control mechanism*
O: user supplied information

(ii) *Deletion of an old user.* It can be implemented by removing that user's *ID* entry from the file system, and the system cannot assign its corresponding eu_i to any new user in the future. Since each user's password has a secret value, du_k, associated with it, this deleted user can no longer use his/her password to impersonate any others.

(iii) *Insertion of a new file.* It can be implemented by updating the passwords of only those users who have access privilege to this new file. Since usually only a few users have access privilege to this new file, it is not necessary to change the majority of the users' passwords.

(iv) *Deletion of an old file.* It can be implemented by removing the corresponding file entry from the file system. The old ef_j associated with it cannot be used again.

(v) *Grant of access right.* It involves two processes. The first one is the privilege verification to determine whether the one who initiates this grant request has the right to grant the privilege. It can be achieved through the verification phase. The second one is to generate a new password to the user who receives this new privilege.

(vi) *Revoke of access right.* It also involves two processes. The first one is the privilege verification to determine whether the one who initiates this revoke request has the right to revoke the privilege. The second one is the most complicated and difficult part in most access control mechanisms. But, in this proposed schemes, it can be achieved just by first deleting the file associated with this access right as in (iv), and then by inserting this file back into the system as in (iii), but only with its remaining access privileges after the revoking.

Example: The access matrix of a simple system with four users and five files is as shown in Fig. 3. The system selects $p = 83$, $q = 107$ and $\alpha = 100$ and calculates $N = 8881$. The system also assigns $ef_j, j = 1, 2, \ldots, 5$, and eu_i, $i = 1, 2, 3, 4$, as shown in Fig. 3. The system will then calculate df_j, $i = 1, 2, \ldots, 5$, du_i, $i = 1, 2, 3, 4$, and assign a password for each user, and one master key for itself, respectively; the results are shown in Table 1.

subject \ object	file 1(3)	file 2(5)	file 3(7)	file 4(11)	file 5(13)
user 1(17)	4	4	1	2	0
user 2(19)	3	3	4	4	0
user 3(23)	1	3	1	3	4
user 4(29)	1	2	1	0	2

Fig. 3 *Access matrix of illustrated example*
* 0: no access; 1: execute; 2: read; 3: write; 4: own

Table 1: Secret numbers, users' passwords and the system's master key of the illustrated example

	du_i	PW_i	t_i
User 1	5113	1809	42879375
User 2	915	7452	118641513375
User 3	4535	3406	99788563875
User 4	3297	4717	88725

$K_{master} = 3088$, $T = 15010$

	df_i
File 1	5795
File 2	3477
File 3	4967
File 4	3951
File 5	5349

When user 3 makes a request to access file 2 with privilege 3, the user will submit $[ID_3, PW_3, t_3, F_2, r] = [ID_3$, 3406, 99788563875, F_2, 3] to the system. The system first checks whether ID_3 is a legitimate user, and then evaluates

$$A = t_3/ef_2^3 = 99788563875/125 = 798308511$$

which is an integer. The system then evaluates the corresponding tokens associated with the requested privilege, based on two different sources: one from the user's submitted password, PW_3, and the other from the system's master key, K_{master}.

$$\begin{aligned} V' &= PW_3^{(A \bullet eu_3 \bmod \phi(N))} \bmod N \\ &= (3406)^{1957} \bmod 8881 \\ &= 8144 \end{aligned}$$

and

$$\begin{aligned} V &= K_{master}^{(T^4/ef_2{}^3 \bmod \phi(N))} \bmod N \\ &= (3088)^{21} \bmod 8881 \\ &= 8144 \end{aligned}$$

Since

$$V' = V$$

this access request is granted.

4.5 Security analysis

Any user ID_i, who tries to access a file with privilege to which he/she is not entitled would have to get at least one secret df_j, $j \in [1, m]$, and then modify his/her password, PW_i, and t_i. However, the computing of df_j without knowing p and q is computationally infeasible. So, even if it is easy for ID_i, who has no privilege r, to access file F_j, to pass the first test by submitting a forged t_i', which is divisible by ef_j^r, since ef_js are public values, it is still hard for ID_i to find a PW_i' which can be used to derive the correct token.

Even though there may be two users with the same access privileges over the files, every password PW_i, $i = 1, 2, \ldots, n$ is unique and cannot be derived from other PW_is since it has the secret du_i embedded in it, which is computationally infeasible to compute. We believe that the security of this new scheme is based on the fact that factoring N into p and q is difficult. We encourage the readers to find a polynomial time algorithm to break this scheme. For more detailed proof of the security, readers can refer to Reference 8.

5 Comparison between conventional approach and our scheme

The most distinct feature in this new scheme is that password authentication and access control can be achieved simultaneously. In the conventional approach, a secure access control mechanism can be fooled by the penetration or bypass of the password authentication mechanism. On the other hand, the performance of a secure password authentication mechanism may also be degraded by the defects in the access control mechanism. However, by integrating both password authentication and access control mechanisms into a single module, our new approach will not have the above problems. Also, each user keeps only a secret password, and the system keeps only its secret master key in this new scheme. it reduces the amount of secret information kept in the system, and therefore enhances the security of the system. Any hacker attempting to crack down the system would find it as difficult as breaking RSA.

Since, in our scheme, user authentication is invoked repeatedly for every data access, some computational overhead may be required. However, it takes only two exponential modular operations for every data access. In comparison with most public key based password authentication schemes [7, 9], our scheme not only has the same computational load but also can provide two protection mechanisms at the same time. In many financial applications, one user may access only one data item each time, for example, the withdrawal of cash from an automatic teller machine. In fact, the more frequently user authentication is performed, the better system security can be expected. In addition, this scheme eliminates the same problem of list searching as found in the conventional approaches.

In this scheme, the values of t_is seem to be large. However, in real situations, each user is allowed to access only a small portion of the resources in the system, so the values of t_is will not be too large. The value of T kept by the system is not a problem either, since it can be reduced as:

$$Y = T \bmod \phi(N)$$

By keeping Y secret in the system, we greatly reduce the storage needed for T. In fact, the total storage of Y and t_is will not be larger than those of access lists, capacity lists, or key-lock pairs of the conventional approaches. More importantly, t_is are now distributed to each user who keeps only his own t_i, and the system does not have to keep the whole access control information.

The user's password being too long to be memorised seems to be another problem. But we know that a short password is very vulnerable to various kinds of attacks, such as exhaustive guess, dictionary search, etc. Therefore, the only way to overcome this problem is to increase the entropy of the passwords. Fortunately a long password can be stored either in a portable diskette or a magnetic card. Besides, there are many other applications, where machines only are involved in the password authentication (for example, in workstations or severs in a computer network). These machines have already been equipped with memory and processors.

6 Conclusion

The integration of user authentication and access control can provide better security for the system. In this paper, we have presented a cryptographic scheme to achieve this goal along with its security analysis. Complexity of time and complexity of space for this scheme are compared with the conventional approach. From the security aspect, it appears to be better than the conventional approach. It also appears to be more efficient for some specific applications.

7 Acknowledgments

Funding for this research has been supported by the Center for Advanced Technology in Telecommunication and Computer Networking, University of Missouri, Kansas City, and the Missouri Department of Economic Development.

8 References

1 CHANG, C.C.: 'On the design of a key-lock-pair mechanism in information protection systems', *BIT*, 1986, **26**, pp. 410–417
2 CHICK, G.C., and TAVARES, S.E.: 'Flexible access control with master keys', *Adv. in Crypto.-Crypto*, 1989, pp. 316–322
3 DENNING, D.E.R.: 'Cryptography and data security' (Addison-Wesley, Reading, MA, 1982)
4 DIFFIE, W., and HELLMAN, M.E.: 'New directions in cryptography', *IEEE Trans.*, 1976, **IT-22**, pp. 644–654
5 GAMAL, T.E.: 'A public key cryptosystem and a signature scheme based on discrete logariths', *IEEE Trans.*, 1985, **IT-31**, pp. 469–472
6 GRAHAM, G.S., and DENNING, P.J.: 'Protection-principle and practice', *Proc. AFIPS SJCC*, 1972, **40**, pp. 417–429
7 HARN, L., HUANG, D., and LAIH, C.S.: 'Password authentication using public-key cryptography', *Computers Math. Applic.*, 1989, **18**, (12), pp. 1001–1017
8 HARN, L., and LIN, H.Y.: 'A cryptographic key generation scheme for multilevel data security', *Computers and Security*, 1990, **9**, (6), pp. 539–546
9 HWANG, T.Y.: 'Passwords authentication using public-key encryption', *Int. Carnahan Conf. On Security Technol.*, 1983, Zurich, Switzerland
10 MERKLE, R.C., and HELLMAN, M.E.: 'Hiding information and signatures in trapdoor knapsack', *IEEE Trans.*, 1978, **IT-24**, pp. 525–530
11 PURDY, G.B.: 'A high security log-in procedure', *Commun. of ACM*, 1974, **17**, (8), pp. 442–445
12 RIVEST, R.L., SHAMIR, A., and ADLEMAN, L.: 'A method for obtaining digital signatures and public-key cryptosystem', *Commun. of ACM*, 1978, **21**, (2), pp. 120–126
13 SALTZER, J.H., and SCHROEDER, M.D.: 'The protection of information in computer systems', *Proc. IEEE*, 1975, **63**, pp. 1278–1308
14 SEELEY, D.: 'Password cracking: a game of wits', *Commun. of ACM*, 1989, **32**, (6), pp. 700–703
15 BIHAM, E., and SHAMIR, A.: 'Differential cryptanalysis of DES-like cryptosystems', *Proc. of Crypto.*, 1990, Santa Barbara
16 BIHAM, E., and SHAMIR, A.: 'Differential cryptanalysis of Snefru, Khafre, REDOC-II, LOKI and Lucipher', *Proc. of Crypto.*, 1991, Santa Barbara
17 WU, M.L., and HWANG, T.Y.: 'Access control with single-key-lock', *IEEE Trans. Software Eng.*, 1984, **SE-10**, (2), pp. 185–191