Computers & Security, 9 (1990) 539–546

Refereed Article

A Cryptographic Key Generation Scheme for Multilevel Data Security

Lein Harn and Hung-Yu Lin

Computer Science Telecommunications Program, University of Missouri-Kansas City, Kansas City, MO, U.S.A.

In 1982, Akl and Taylor proposed an elegant solution to the partially ordered multilevel key distribution problem, using a cryptographic approach. Since then, continuing research has been conducted to try to realize and simplify their scheme. Generally speaking, there are two problems associated with their scheme. First, a large value associated with each security class needs to be made public. Secondly, new security classes are not permitted to be added into the system once all the security keys have been issued. Our paper presents a very similar approach. But, instead of using the top-down design approach as in their scheme, our scheme is using a bottom-up key generating procedure. The result is that the published values for most security classes can be much smaller than in their scheme. This property becomes more obvious for a broad and shallow hierarchical graph. In addition, our scheme can accommodate the changes of adding new security classes into the system.

Keywords: Cryptographic scheme, Multilevel data security, Key distribution, Partially ordered hierarchy, RSA scheme.

1. Introduction

The multilevel data security problem originally exists in military and government departments as well as some private corporations where classified data management is necessary. Now, because of the increase in computing resources, it is more frequently found in applications such as database management [1–3], computer networks [4–7], and operating systems [8, 9].

The multilevel security problem exists in many organizations where a hierarchical structure of data sensitivity and user privilege coexists. Government and military organizations are the classic examples of such hierarchies [10]. There are also examples in commercial environments. For instance, a corporate hierarchy may be organized in a tree structure, with top management at the root and security classes corresponding to divisions, departments, and projects at successive levels of the tree. A manager of a division has clearance for the security class of that division and, thereby, is authorized to access information in all departments and projects within that division. Members of a project team, on the other hand, are cleared only for that project and will be unable to access information concerning other projects, including those within the same department. A totally different application environment would be a computer running in a multilevel secure mode having users with different access rights as well as objects of various sensitivity levels.

To make the rest of our paper more precise in describing these relationships, we define the following basic terms:

- Let $\mathscr{U}$ be the set of users in the computing environment. We define the set of security classes $\mathscr{T}=\{C_i\}$ of the computing environment to be a partition of $\mathscr{U}$, *i.e.* $\mathscr{T}=\{C_i\}$

  for $1 \leqslant i \leqslant m$, such that $\bigcup_{i=1}^{m} C_i = \mathscr{U}$ and $C_i \cap C_j = \emptyset$

  for $i \neq j$.

- Further, we assume that the set of security classes $\mathscr{T}$ is ordered in a hierarchy by the relation $\leqslant$, where $C_j \leqslant C_i$ means that C_j is subordinate to C_i.

- We define a bijection *key*: $\mathscr{T} \rightarrow \mathscr{K}$ for $\mathscr{K}=\{K_i\}$, a set of keys, such that *key* $(C_i)=K_i$, for $1 \leqslant i \leqslant m$. Thus, for all u_1, $u_2 \in C_i$, u_1 and u_2 share K_i and all data secured under K_i.

- Let $\mathscr{S}(C_i)$ be the set of all subordinate classes of a given security class C_i. That is, $C_j \in \mathscr{S}(C_i)$ iff $C_j \leqslant C_i$.

- Also, we define $\overline{\mathscr{S}(C_i)} = \mathscr{T} - \mathscr{S}(C_i)$.

- Let $L_i \in \mathbb{Z}$ represent the level of the security class C_i in $\mathscr{T}$.

With this basic terminology, the multilevel security problem is now formally defined to be the problem of securing the data of all security classes in a partially ordered set $\mathscr{T}$ such that C_i has access to data accessible to users in C_j iff $C_j \leqslant C_i$. This problem can be solved by the encryption of the data. The set of keys $\mathscr{K}$ is used to secure the classes $\mathscr{T}$, such that each user $u \in C_i$ holds a subset of keys $\mathscr{K}$ so as to be able to access his/her own data and that of subordinate users.

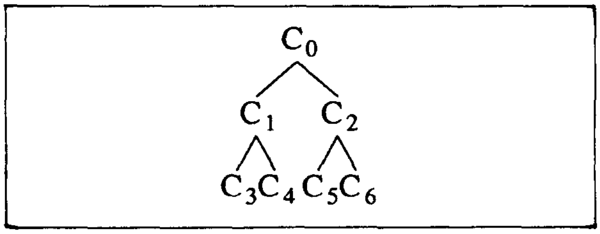

Fig. 1. A multilevel security hierarchy.

TABLE 1 The set of keys required to be held by each class in hierarchy given in Fig. 1.

Security class	*Keys held*
C_0	$K_0, K_1, K_2, K_3, K_4, K_5, K_6$
C_1	K_1, K_3, K_4
C_2	K_2, K_5, K_6
C_3	K_3
C_4	K_4
C_5	K_5
C_6	K_6

This quantity of keys is awkward to administer and may itself become a security hazard. This is called the key management problem of multilevel security. For example, given the special tree multilevel hierarchy shown in Fig. 1, in order to retrieve the data encrypted under each user's own key or subordinates' keys, that user has to hold a set of keys as shown in Table 1.

It is shown in Table 1 that higher privileged users, entitled to retrieve more secret information, are required to hold more keys. The large number of keys held by users at the higher level is a disadvantage, especially in systems with large numbers of security classes. Many security issues are involved when the information (or key) needs to be stored in secret. Storage for the keys is one concern because the secret key is sizable because of the security requirement. The fact is that the more keys need to be held, the more risks are involved; keys can be lost or stolen. Therefore the goal is to find a mechanism such that each user needs to hold only one key and that user is able to use that key to retrieve all the information to which he/she is entitled, while retaining a secure system.

2. Review of the Akl/Taylor Scheme

The first attempt to solve the key management problem employing cryptographic techniques was proposed in 1982 by Akl and Taylor [11]. They assume a communication system where every user belongs to one of a set of disjoint security classes and periodically receives data from an authority. The set of classes is partially ordered by the relation $\leqslant$, where $C_j \leqslant C_i$ means that any user $u \in C_i$ can have access to information destined to any user $u' \in C_j$.

The problem is to design a scheme such that an object x broadcast by $u \in C_m$ and addressed to $u' \in C_j$ is accessible to $u'' \in C_i$ if and only if $C_j \leqslant C_i$. Akl and Taylor assume the existence of a key center (KC), which is responsible for key generation and key distribution. Each user in the network receives a single secret key generated by KC and uses this key to derive the secret keys of the user's subordinates. When $u \in C_i$ wishes to broadcast a message x to $u' \in C_j$, he/she first enciphers it under K_j to obtain

$$x' = E_{K_j}(x)$$

and then broadcasts $[x', j]$. Only users in possession of K_j (or who can derive K_j) will be able to retrieve x by calculating

$$x = D_{K_j}(x')$$

The key generation algorithm proposed by Akl and Taylor to solve this problem is given below.

2.1. Key Generation Procedures

Step 1: Assign each security class C_i an associated distinct prime P_i, such that $P_i \neq P_j$ if $C_i \neq C_j$.

Step 2: Generate t_i for each security class C_i, where t_i is the following function of the primes from the previous step,

$$t_i = \prod_{C_j \in \overline{\mathcal{S}(C_i)}} P_j$$

Then publish t_i.

Step 3: The KC chooses a secret pair of large prime numbers p and q, with product $M = p \cdot q$, and a random secret key K_0, $2 \leqslant K_0 \leqslant M-1$, $\gcd(K_0, M) = 1$. M is made public.

Step 4: The secret keys for all security classes $C_j \in \mathcal{T}$ can now be computed by KC as follows:

$$K_j = K_0^{t_j} \bmod M.$$

2.2 Key Derivation Procedures

User $u \in C_i$ can derive the key K_j by the formula

$$K_j = K_i^{t_j/t_i} \bmod M,$$

iff $C_j \leqslant C_i$

According to the rules of this scheme of assigning primes, the public integers t_i have a special feature which makes the key derivation possible, namely

t_j is divisible by t_i if and only if $C_j \leqslant C_i$

This statement is true because $\overline{\mathcal{S}(C_i)} \subseteq \overline{\mathcal{S}(C_j)}$ if $C_j \leqslant C_i$, thus t_j is divisible by t_i. Conversely, $\overline{\mathcal{S}(C_i)} \nsubseteq \overline{\mathcal{S}(C_j)}$, if $C_j \nleqslant C_i$, hence t_j is not divisible by t_i. This point is extremely important, since the security of the scheme relies on it. This is to say that if t_j/t_i is an integer, the key derivation works successfully and C_i has privilege over C_j; otherwise it fails, which implies C_i has no privilege over C_j.

However, two problems arise in the practical implementation of this scheme. First, since each security class is assigned a distinct prime number P_i, and the corresponding integer t_i of each class is a product of all primes which are not subordinate to this security class in the hierarchy, it must be evaluated for each sensitivity level in advance. The size of these integers is proportional to the number of users in the network; therefore, when the number of users in the network becomes large this method becomes impractical, since t_i grows dramatically with the increasing number of security classes. Secondly, it will violate the security requirement whenever a new security class is added into the system and it becomes the subordinate of any existing security class. In other words, it is impossible to expand the system whenever all the security keys have been issued.

Akl and coworkers have consistently attempted to solve the sizable t_i problem. First, in 1983, MacKinnon and Akl [12] proposed two new algorithms to reduce the value of t_i; however, the result was still not satisfactory. Later, in 1985, MacKinnon *et al.* finally solved the mystery and found an optimal algorithm for assigning t_i [13, 14]. Unfortunately, t_i is still sizable. Therefore there is no other way to reduce the value of t_i and the sizable t_i problem is thus left unsolved.

While Akl and Taylor have solved the multilevel security problem for the general case of a partially ordered hierarchy, there are other researchers looking for solutions for the special case of a tree hierarchy. In 1987, Sandhu [15] proposed an ID-based scheme for solving the key generation problem in a tree-structured multilevel data security environment. Based on his scheme, each user's secret key is calculated from his/her own ID and his/her supervisor's secret key through a one-way function. In this way, no extra public information is needed for the key derivation. Another important advantage is that the insertion of new security classes can be easily handled. However, it requires computational overhead in deriving keys.

In this paper, we propose a key generation scheme by eliminating the sizable t_i problem for certain hierarchies in comparison to Akl and Taylor's scheme. Generally speaking, instead of using a top-down design approach, as in all the existing algorithms, we present a bottom-up key generating scheme. The result is that the published t_i values for most security classes are much smaller than Akl, Taylor and MacKinnon's scheme. In addition, our scheme can accommodate the changes of adding new security classes into the system.

3. Our Scheme for a Totally Ordered Hierarchy

This is the simplest multilevel hierarchy, such that for any two security classes C_1 and C_2, where $C_1 \neq C_2$, either $C_1 \leqslant C_2$ or $C_2 \leqslant C_1$. Based on the difficulty of solving "factoring a product of two large primes," the following algorithm to assign each security class a secret key is proposed.

3.1 Algorithm

3.1.1 Key Generation Procedures

Step 1: The KC chooses and keeps three parameters for the key generation. These three parameters are two large 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.* $\gcd(\alpha, n) = 1$. The KC also calculates a secret value, d, such that $d \cdot 3 \bmod \phi(n) = 1$, where $\phi(n)$ is the Euler's totient function of n. In other words, $d = 3^{-1} \bmod \phi(n)$.

Step 2: The KC generates a set of keys $K = \{K_i\}$ for all the security classes such that

$$K_i = \alpha^{d^{L_i}} \bmod n,$$

where L_i is the level of C_i and which is counted from the lowest level.

Thus, each user $u \in C_i$ holds only one key K_i.

3.1.2 Key Derivation Procedures

If user $u_i \in C_i$ wishes to access the data of a user $u_j \in C_j$ and $C_j \leqslant C_i$, then u_i can derive $key(C_j) = K_j$ as

$$K_j = K_i^{3^{L_i - L_j}} \bmod n$$

where L_i is the level of C_i and L_j is the level of C_j. Now, Theorem 1 below, proves that the key derivation is correct.

[Theorem 1] Given C_i and C_j such that $C_j \leqslant C_i$ and given the keys K_i and K_j generated by the key generation procedures, so that *key* $(C_i) = K_i$ and *key* $(C_j) = K_j$, and given that the key K'_j is generated by the key derivation procedure, then $K_j = K'_j$.

Proof: $K'_j = K_i^{3^{L_i - L_j}} \bmod n$

where $C_j \leqslant C_i$, and hence $L_i > L_j$

$= (\alpha^{d^{L_i}} \bmod n)^{3^{L_i - L_j}} \bmod n$

$= \alpha^{d^{L_i} \cdot 3^{L_i - L_j}} \bmod n$

$\alpha^{(d^{L_i} \cdot 3^{L_i})(3^{-L_j})} \bmod n$

where $d \cdot 3 \bmod \phi(n) = 1$

$= \alpha^{3^{(-1)L_j}} \bmod n$

where $3^{(-1)} \bmod \phi(n) = d$

$= \alpha^{d^{L_j}} \bmod n$

according to the key generation procedures

$= K_j$ QED

3.2 Example

Given the totally ordered hierarchy in Fig. 2, where $C_4 \leqslant C_3 \leqslant C_2 \leqslant C_1 \leqslant C_0$, Figure 3 shows the keys assigned to each security class associating to our scheme.

C_0	level 5
C_1	level 4
C_2	level 3
C_3	level 2
C_4	level 1

Fig. 2. A totally ordered security hierarchy.

$\alpha^{d^5} \bmod n$

$\alpha^{d^4} \bmod n$

$\alpha^{d^3} \bmod n$

$\alpha^{d^2} \bmod n$

$\alpha^{d} \bmod n$

Fig. 3. Secret keys assigned for the security hierarchy given in Fig. 2.

3.3 Security Analysis

The security of this proposed algorithm is equivalent to that of the RSA scheme [16] whose security at best is based on the computational difficulty of factoring a product of two large primes. As in the RSA scheme, the key center needs to pick 3 relatively prime to $\phi(n)$ in the interval $[1, n-1]$. This is assured by choosing p and q to be safe primes. The corresponding inverses $d = 3^{-1} \bmod \phi(n)$, and $\phi(n)$, are kept secret. It is computationally infeasible for any user (or outsider) to determine the secret key d either by solving the equation $3 \cdot d \bmod \phi(n) = 1$ or by solving the equation $K_i = \alpha^{d^{L_i}} \bmod n$ (which is equivalent to breaking the RSA scheme).

4. Partially Ordered Hierarchy

4.1 Algorithm

4.1.1 Key Generation Procedures

Step 1: The KC chooses and keeps three parameters for the key generation. These three parameters are two large 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: The KC assigns each security class, C_i, a distinct prime, e_i, and makes these primes publicly available. It can start to assign these primes to security classes from the bottom of the graph and select the smallest odd prime available.

Step 3: The KC calculates the multiplicative inverse, d_i, for each class. That is $d_i = e_i^{-1} \bmod \phi(n)$.

Step 4: The KC calculates a set of public values $t = \{t_i\}$ and security keys $K = \{K_i\}$ for all security classes in the hierarchy such that

$t_i = \Pi e_j$ and $K_i = \alpha^{\Pi d_j \bmod \phi(n)} \bmod n$, for all j such that $C_j \in \mathscr{S}(C_i)$.

4.1.2 Key Derivation Procedures

If user $u_i \in C_i$ wishes to access data of user $u_j \in C_j$ and $C_j \leqslant C_i$, then u_i can derive *key* $(C_j) = K_j$ as

$K_j = K_i^{t_i/t_j} \bmod n$.

Now, Theorem 2 below, proves that the key derivation is correct.

[Theorem 2] Given C_i and C_j such that $C_j \leqslant C_i$ and given the keys K_i and K_j generated by the key generation procedures, so that *key* $(C_i) = K_i$ and *key* $(C_j) = K_j$, and given that the key K'_j is generated by the key derivation procedure, then $K_j = K'_j$.

Proof: $K'_j = K_i^{t_i/t_j} \bmod n$

where $C_j \leqslant C_i$, and hence $t_i > t_j$

$= (\alpha^{\Pi d_k})^{(\Pi e_k / \Pi e_m)} \bmod n$

for all k and m such that $C_k \in \mathscr{S}(C_i)$ and $C_m \in \mathscr{S}(C_j)$

$= (\alpha^{\Pi d_m})^{(\Pi e_s \cdot d_s)} \bmod n$

for all s such that $C_s \in \mathscr{S}(C_i) - \mathscr{S}(C_j)$

$= (\alpha^{\Pi d_m}) \bmod n$

where $(\Pi e_s \cdot d_s) \bmod \phi(n) = 1$

$= K_j$ QED

4.2 Security Analysis

The relation between the personal secret keys belonging to two unrelated distinct users u_i and u_j is very simple. For example, user u_j's key cannot be calculated from user u_i's key K_i since user u_i cannot discover u_j's system secret key d_j. Similarly, the conspiracy of all children keys cannot discover the ancestors' keys since each ancestor's key includes his/her own system secret key d_j. All of these attacks run up against the problem detailed in the previous section, and so discovering some key to which one or more users is not entitled would be equivalent in difficulty to breaking the RSA scheme. In summary, as long as all the system secret keys $\{d_i\}$, p and q, are kept secret, conspiracy attacks cannot succeed, and individual users of the system are bound to only what they are supposed to know.

5. Comparisons Between Akl and Taylor's Scheme and Our Scheme

Let us use the following example taken from Akl and Taylor's original paper [11] to illustrate the difference between these two schemes.

5.1. Example

Given the partially ordered graph has six security classes, C_i, $i = 1, 2, \ldots, 6$, as shown in Fig. 4, Table 2 shows the keys assigned to each security class associating with Akl and Taylor's scheme and our scheme.

There are several differences between Akl and Taylor's scheme and our scheme.

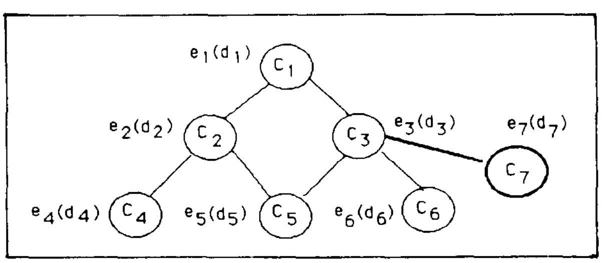

Fig. 4. A partially ordered security hierarchy.

TABLE 2 The set of keys required to be held by each class in hierarchy given in Fig. 4.

	Akl and Taylor's scheme	*Our scheme*
K_1	$\alpha\ (t=e_1)$	$\alpha^{d_1d_2d_3d_4d_5d_6}\ (t=e_1e_2e_3e_4e_5e_6)$
K_2	$\alpha^{e_1e_3e_6}\ (e_1e_3e_6)$	$\alpha^{d_2d_4d_5}\ (e_2e_4e_5)$
K_3	$\alpha^{e_1e_2e_4}\ (e_1e_2e_4)$	$\alpha^{d_3d_5d_6}\ (e_3e_5e_6)$
K_4	$\alpha^{e_1e_2e_3e_5e_6}\ (e_1e_2e_3e_5e_6)$	$\alpha^{d_4}\ (e_4)$
K_5	$\alpha^{e_1e_2e_3e_4e_6}\ (e_1e_2e_3e_4e_6)$	$\alpha^{d_5}\ (e_5)$
K_6	$\alpha^{e_1e_2e_3e_4e_5}\ (e_1e_2e_3e_4e_5)$	$\alpha^{d_6}\ (e_6)$

(1) In Akl and Taylor's scheme, each public t_i is the product of primes associated with those security classes which are supervisors of C_i or are not related to C_i. But in ours, t_i is the product of primes associated only with its subordinates. So, in most cases, the size of public information (*i.e.* t_i's) is much smaller in our scheme than in Akl and Taylor's. For a broad and shallow hierarchy, this difference becomes more obvious.

(2) By way of comparison, if in Akl and Taylor's scheme, the key center selects the initial secret root key, K_0, to be equivalent to the last-generated root key in our scheme, then all the secret class keys generated in these two schemes will become identical. In other words, our scheme is actually a complementary scheme of Akl and Taylor's scheme which provides an alternative way for generating hierarchical keys from an opposite direction.

(3) Consider the case where after all keys have been issued, a new security class, C_7, needs to be added into the hierarchy as shown in Fig. 4. In Akl and Taylor's scheme, this new key would be generated as $K_7 = \alpha^{e_1e_2e_3e_4e_5e_6}$. However, it violates the security requirement because K_7 can be obtained from any of the old keys, K_i, $i \in [1, 6]$. Therefore a completely new key generating procedure needs to be performed for all security classes. However, in our scheme, only those keys which have access privilege to this new security class need to be updated. In this example, there are only two keys, K_1 and K_3, which need to be updated. The remaining majority of the keys do not need to be changed.

6. Conclusion

We have proposed a new systematic solution to the multilevel key generation problem. The results show that our scheme (a) is more efficient in the memory utilization since it needs less space to keep public information and (b) can handle new user's insertion without changing all keys.

References

[1] Committee on Multilevel Data Management Security. Multilevel Data Management Security, *Tech. Rep., Air Force Studies Board, National Research Council*, 1982.

[2] D. E. Denning, S. G. Akl, M. Morgenstern and P. G. Neumann, Views for multilevel database security, *Proc. 1986 IEEE Symp. on Security and Privacy, Oakland, CA, April 7–9, 1986*, pp. 156–172.

[3] D. E. Denning, Cryptographic checksums for multilevel database security, *Proc. 1984 IEEE Symp. on Security and Privacy, Oakland, CA, April 29–May 2, 1984*, pp. 52–61.

[4] R. Nelson and J. Jarzembowski, Multilevel security: an overview and new directions, *Proc. 1977 IEEE Symp. on Trends and Applications, Gaithersburg, MD, May 19, 1977*, p. 41–48.

[5] J. McHugh and A. P. Moore, A security policy and formal top level specification for a multi-level secure local area network, *Proc. 1986 IEEE Symp. on Security and Privacy, Oakland, CA, April 7–9, 1986*, pp. 34–39.

[6] D. McCullough, Specifications for multi-level security and a hook-up property, *Proc. 1987 IEEE Symp. on Security and Privacy, Oakland, CA, April 27–29, 1987*, pp. 161–166.

[7] W. P. Lu and M. K. Sundareshan, A model for multilevel security in computer networks, *Proc. 1988 INFOCOM, New Orleans, LA, March 27–31, 1988*, pp. 1095–1104.

[8] D. E. Bell and L. J. Lapadula, Secure computer system: unified exposition and multics interpretation, Tech. Rep., MTR-2997, March 1976 (MITRE Corporation, Bedford, MA).

[9] L. J. Fraim, Scomp: a solution to multilevel security problem, *IEEE Computer*, (July 1983) 26–143.

[10] U.S. Department of Defense, National Computer Security Center, Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria, 1987.

[11] S. G. Akl and P. D. Taylor, Cryptographic solution to a multilevel security problem, *Proc. Crypto-82, Santa Barbara, CA, August 23–25, 1982*, pp. 237–250.

[12] S. MacKinnon and S. G. Akl, New key generation algorithms for multilevel security, *Proc. 1983 IEEE Symp. on Security and Privacy, Oakland, CA, April 25–27, 1983*, pp. 72–78.

[13] S. J. MacKinnon, P. D. Taylor, H. Meijer and S. G. Akl, An

optimal algorithm for assigning cryptographic keys to access control in a hierarchy, *IEEE Transactions on Computers*, C-34(9) (September 1985) 797–802.

[14] S. G. Akl and P. D. Taylor, Cryptographic solution to a problem of access control in a hierarchy, *ACM Transactions on Computer Systems*, 1(3) (August 1983) 239–248.

[15] R. S. Sandhu, Cryptographic implementation of a tree hierarchy for access control, *Information Processing Letter*, 27 (1988) 95–98.

[16] R. L. Rivest, A. Shamir and L. Adleman, A method for obtaining digital signature and public-key cryptosystems, *Communication ACM*, 21 (1978) 126.

Lein Harn was born in Taipei, Taiwan, in 1954. He received the B.S. degree from the National Taiwan University in 1977, the M.S. degree from the State University of New York, Stony Brook, NY, in 1980, and the Ph.D. degree from the University of Minnesota, Minneapolis, MN, in 1984, all in electrical engineering.

From 1981 to 1984 he was a Research/Teaching Assistant and was involved in research on signal detection and digital filtering in the Department of Electrical Engineering at the University of Minnesota. Since 1984 he has been an Assistant Professor in the Department of Electrical and Computer Engineering, University of Missouri at Columbia, and the Department of Computer Science, University of Missouri at Kansas City, MO. From 1986 to 1987 he was a Visiting Associate Professor at the National Cheng Kung University, Taiwan, R.O.C. Currently he is an Associate Professor at the Computer Science Telecommunications Program, University of Missouri at Kansas City, MO. His research interests include digital filter design, multidimensional digital signal processing, data security, cryptography, and VLSI design.

Hung-Yu Lin was born in Yunling, Taiwan, in 1963. He received the B.S. degree from the Department of Computer Science, Tunghi University, Taiwan. From 1987 to 1988 he was a system engineer at the Nanyang Auto Inc., Taiwan. Currently he is a graduate student at Computer Science Telecommunications Program, University of Missouri at Kansas City, MO. His research interests include data security and cryptography.