International Journal of Control and Automation
Vol. 7, No. 11 (2014), pp. 49-58
http://dx.doi.org/10.14257/ijca.2014.7.11.06

Design and Development of a Security Kernel in an Embedded System

Liu Shian

Department of Information and Communication Engineering, Guangzhou Maritime Institute, Guangzhou, China
liushiangz@126.com

Abstract

Dynamic strategy-based security Kernel is very important in an embedded system. It is the core of a secure computing environment that can be implemented in the form of a hardware component installed in a computer or network topology, a software implementation, or a firmware system installed in a computer microchip, or in an embedded system. With increasing dependence on embedded systems, their reliability and security become more and more of an issue. This paper designs and develops a security Kernel prototype system which is named as SK-I. This prototype is very generic since it could be easily inserted to other systems or platforms. It is based on the safety identification, access control, user authority verification, and authority mechanisms. SK-I has been tested under difference dimensions so that the functionalities and performance are examined. From the testing, it is observed that create and delete operation costs a lot and the decision buffer plays an important role in enhancing the performance of an embedded system.

Keywords: *Security; Kernel; Embedded System; CRTOSII*

1. Introduction

Dynamic strategy-based security Kernel is very important in an embedded system. The reliability of an embedded system heavily depends on the architecture of the security Kernel [1]. A security Kernel of key security-related statements that (a) works as a part of an operating system to prevent unauthorized access to, or use of, the system and (b) contains criteria that must be met before specified programs can be accessed. A security kernel is an essential nucleus of a computer or network security implementation [2]. It is the core of a secure computing environment that can be implemented in the form of a hardware component installed in a computer or network topology, a software implementation, or a firmware system installed in a computer microchip, or in an embedded system. By whatever it means, the Kernel becomes the central location for establishing access permissions for a computer or network's resources.

One early security Kernel implementation was a virtual machine monitor developed in the 1970s for the Digital Equipment Corporation® (DEC®) virtual address extension (VAX) computer systems [3]. The computer could be set up into multiple virtual machines that could each run a different operating system, accessing to different resources and with different security protocols. In this instance, the security Kernel resided on the real machine where the virtual machines were established and handled access control for the different virtual machines which could then have varying levels of security.

Embedded systems are part of a larger system that is primarily in an electronic [4]. In industrialized countries, they outnumber people by about an order of magnitude. This includes cell phones, PDAs, entertainment devices, cars, washing machines, smart cards, broadband modems, and many more [5,6]. With increasing dependence on embedded systems, their reliability and security become more and more of an issue. For example, cell phones and PDAs are used to perform financial transactions, which mean that they

ISSN: 2005-4297 IJCA

are trusted with account access codes. Embedded devices also store increasing amounts of sensitive personal data, from address books to medical data. Hence, the security of such systems is a serious concern.

This paper designs and develops a security Kernel prototype system which is named as SK-I. This prototype is very generic since it could be easily inserted to other systems or platforms. It is based on the safety identification, access control, user authority verification, and authority mechanisms. For example, this prototype system is able to support some security strategies like MLS, DTE, and RBAC. Additionally, the prototype system includes the strategic language and compile unit. Based on the language and units, it is easy for end-users to develop the management tools and provide the basis for updating the dynamic strategies.

The paper is organized as follows. Firstly, Section 2 briefly introduces the SK-I prototype system. Secondly, Section 3 demonstrates the design methods, used technologies, and development considerations. Finally, Section 4 reports on the testing in terms of functionalities and performance. The results are analyzed and discussed in this section as well. Finally, Section 5 concludes this paper by giving our findings and future work.

2. SK-I

SK-I (Security Kernel - I) is designed for the embedded real-time operation system. It is based on the excellent characteristics such as high reliability, easy-to-implement, and flexible-to-transfer. SK-I uses C and assembly language to develop. It is compiled and debugged under GCC and GDB in GNU. SK-I is a typical security item which is a modularitibility, multi-layer, and real-time system. It has several components: hardware concrete unit, core unit, server unit, and interface unit.

Hardware concrete unit is designed for connecting the hardware where differences may exist. After the connection, this unit sends the information to the upper level system which is able to achieve transparency on hardware layer. Thus, the transferability could be enhanced of the entire prototype like CPU supporting, BSP and drivers *etc*.

Core unit is the key component of the SK-I system, providing the basis of multi-tasks execution environment. This unit is responsible for managing the tasks, user extension, memory control, communication, synchronization, interruption, clock management, and I/O management *etc*.

Sever unit uses the configurable components to extend the function in SK-I. It aims to enhance the Safety service by using different components and file system modules under complex environments. For instance, the sever unit may use the file system module, network control, and GUI module. These modules are (re)configurable so that the unit is easily implemented and designed.

Interface unit is designed for communicating with the other systems and modules. It provides several standard interfaces for this purpose such as Kernel API, PosixAPI, OSE API and so on. Thus, different situations may use different strategies to handle the interface design and development.

3. Design and Development Considerations

3.1. Design Considerations

The design of SK-I aims to enhance the security ability so that the embedded system is able to perform well when using in different applications. Thus, it is necessary to reduce the coding procedure, which can improve the coupling of different modules. As a result, the original functionalities of the system could be remained. The SK-I is developed through extensible methodologies. For example, the SK-I could be added by integrating

the security units such as strategic verification unit, strategic database that is able to be updated, *etc*. Several considerations will be taken into account when design and develop the SK-I.

Firstly, the security labels are very important so that they named as SID in SK-I are designed for the central control of the corresponding entities within the system. The SID is the basis for deciding the compulsory access control. For an embedded system, all the interoperability of various objects should be verified by checking their SIDs. That means the objects are bound with a unique SID thus, the operations could be detected such as tasks, file system, message queue, signal, events, network ports, and so on.

Secondly, the user verification could be carried out before executing any operations and behaviors. The verification will identify the authorized users so that typical hackers will be rejected. To this end, the verification service has to keep the data, which are used for verifying individual authorities and the access domain within the embedded systems. The verification mechanisms play an important role in keeping the security of an embedded system. As a result, the mechanisms in SK-I are based on the concepts of safety level and authority distribution methods, from where entities with certain behaviors external the security kernel are able to guarantee the reliability and confidentiality. Additionally, they allow the authorized users to act their powers under certain controls, aiming to verify the local end-users in an embedded system.

Thirdly, the design and development of compulsory access control (CAC) should be based on the control strategies, which will be used for managing the file systems and threads. The objects should be tagged with SIDs that are categorized into different levels. Therefore, CAC could execute the logics strictly, aiming to achieve the security control in the embedded system.

Fourthly, SK-I is able to establish, keep, and protect all the objects that are concerned in the embedded system so as to avoid unauthorized access and deteriorization. In order to achieve this purpose, the data related to this function should be encrypted and decrypted. Additionally, the security log should be well designed so that it could be traced and tracked when there are some emergencies and unsafe events.

Fifthly, SK-I considers the multiply security mechanisms such as TE, RBAC, MLS, and ACL. In such case, the update could be carried out through online approach. It enhances the operability of the security Kernel.

Finally, the source codes of SK-I are relatively independent in order to achieve the adaptive safety control. The codes are tight in logics so that they are easily evaluated. A code set is used for realizing all the functionalities, aiming to ease the testing when there are some modifications of the security mechanisms as well as to keep the adaptive security.

3.2. Development of SK-I

The development of SK-I is divided into several parts, which are security API, strategic sever, access monitor, decision buffer, verification service, authority service, and strategic language & tools.

3.2.1. Security API: It is responsible for controlling the access of every resource in the embedded system. It is regarded as a convertor, which is able to transfer the operation system to access monitor so as to execute the authority verification. If the end-user cannot fetch the security API, it is impossible for them to get involve into this part. Thus, security is the basis of the access control.

Intercept control requirements for safety related entities allocate storage space or functional behavior before implementation in order to avoid unnecessary operations. Meanwhile, in the embedded system, when the object is created successfully, it is necessary to assign the appropriate security attributes (mainly SID). Since the codes of security API is frequently modified, safety nuclear is developed as a configurable

module. In order keep the functionality and extensibility, we define the macro CONFIGURE_SECURITY_KERNEL as switching the security code in the API implementation system (including data structures). The security Kernel of the operating system is also dependent modification to the source where the user profile is the macro settings to determine whether binding security module [7].

Take the API tasks for example, the SK-I structure is developed for controlling the granularity of the embedded system. Thus, the monitored objects like tasks, files, and IPC should have the definition of security attributes. The API tasks is defined as follows:

```
struct task_security_struct
 {
Thread_Control *        task;
 struct list_queue      list;
SK_security_id          osid;
SK_security_id          sid;
DC_entry_ref_t          sdcr;
};
```

3.2.2. Strategic Server: It closely relates to the SK-I because it enables the loading and update security strategy. It also decides the access control and manages the SID and security attribute. The main functionalities are access control. When the access monitor sends the requirements, the matched options will be selected from the decision buffer. If there is not matched result, the strategic server will make the decisions. The decision-making algorithm is designed as follows:

```
Boolean AccessCheck(source_sid, target_sid, requested_mode)
Var allowed: access_mask= Æ ;
Allowed the access mode of the DTE verification;
If(!requrested_mode||(requested_mode & allowed)!=reqested_mode)
      Access reject;
Else
      Access enter;
```

After all the permission of strategic verification under the access mode, the strategic server will return the access. Since the DTE and RBAC verification algorithms are simple, this paper uses MLS algorithm for more complex situations. MLS algorithm contains two parts: low and high level, aiming to determine the isolated domain. There are several access modes in MLS [8]. They are read, readby, write, and writeby. All the operations from AIP could be mapped into these modes so that the security Kernel could be achieved in the embedded system. The algorithm could be expressed as follows:

```
MLS_Verifier (Source, Target, Allowed)
If (source.high!=target.high)
      If!(source.high>=target.high)
            Allowed=allowed&~(target->mlsperms.read);
if !( source. high <=target.high)
allowed = allowed & ~(target->mlsperms.readby);
if !((source.low <= target.low = target.high) or
(source.low <= target.low <= target.high <= source.high))
allowed = allowed& ~(tatget->mlsperms.write);
```


if!((target.low <= source.low = source.high) or
(target.low <= source.low <= source.high <= target.high))
allowed = allowed& ~(tatget->mlsperms.writeby);
return allowed;

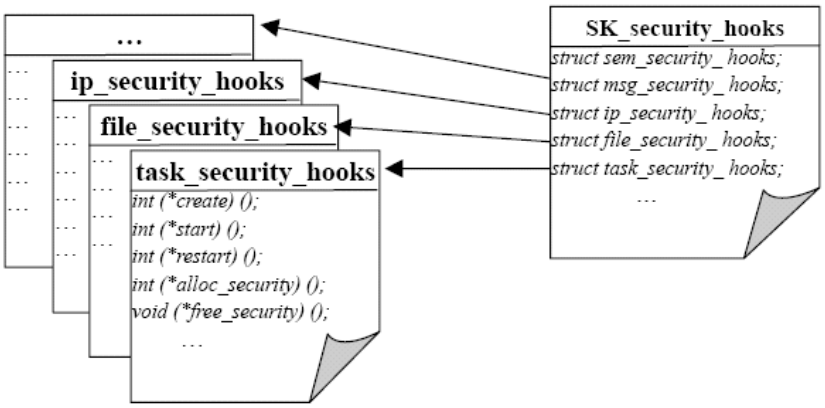

Figure 1. Data Structure of Security Hooks

3.2.3. Access Monitor: It is responsible for monitoring corresponding entities, accepting the requirements, and executing the establishment of strategies. In SK-I, it is inserted into process management service, network service and file service, twining with API for transferring the requirements related to the security. Through the interface between the strategic server and access monitor, it is able to provide the parameters for verifying the data and give response to the requirements.

Access monitor uses security hooks for obtaining the operations from the API layer and sending the requirements of access authorities. Security hooks is a queue point, which is a global function. It is used for verifying the operations related to the security. The design of security hooks is shown as Figure 1.

3.2.4. Decision Buffer: It is responsible for improving the facilities within SK-I. If all the requirements from the security entities are sent to strategic server, it is costful for an embedded system. Thus, SK-I uses the decision buffer for reducing the costs. The requirements of the successful security concerns are kept with the form of <SID, SID, access mode list>. When the access monitor is going to get the requirements, it will query the decision buffer for matching the target items. If matched, the results will be returned. Otherwise, both the SID and requirements will be sent to the strategic sever. In SK-I, a buffer table DC_entry_T is used for storing the definition of attributes. It is shown as follows:

typedef struct DC_entry {
SK_security_id ssid;
SK_security_id tsid;
SK_security_class tclass;
unsigned32 allowed;
unsigned32 decided;
unsigned32 audit;
unsigned32 used;
} DC_entry_T;

The decision buffer realizes the support for recall functions based on the key degree. Thus, the decision buffer is able to provide more services and query functions given different situations.

3.2.5. Verification Service: It is used for recording the security activities within an embedded system. In SK-I, this service keeps the access records such as access requirements, access decisions, strategic data loading, operations on the system, *etc*. If there are some emergencies and system breakdown, this service is able to locate, check, and detect the details of the accident, providing real-time information, reliable evidence and son on. Thus, the unsuitable events will be alarmed and the location, process, and duty could be traced and tracked.

SK_CRTOS usually contains the following information and data: event date and time, categories of the executing entities, object ID and SID, event type, labels, *etc*. Verification log is stored in a binary file, which are with high priority so that they are protected by the compulsory access control system [9]. Through the related management tools, these files could be interpreted and displayed to the end-users. The data structure of the system log is presented as follows:

```
Struct SK_auditLog_arg {
SK_security_id ssid;
SK_security_id tsid;
struct access_event ae;
struct log_time time;
unsigned32 logid;
}
```

The verification service checks most of the points such as shell access, sub-systems, decision buffer, strategic server, *etc.*, all the check results will be formed as the log which is stored in the system for historic query.

3.2.6. Authority Service: It is used for ensuring the login security when an end-user tries to get into the system. In SK-I, the checking logics is used. The detail is shown in Figure 2.

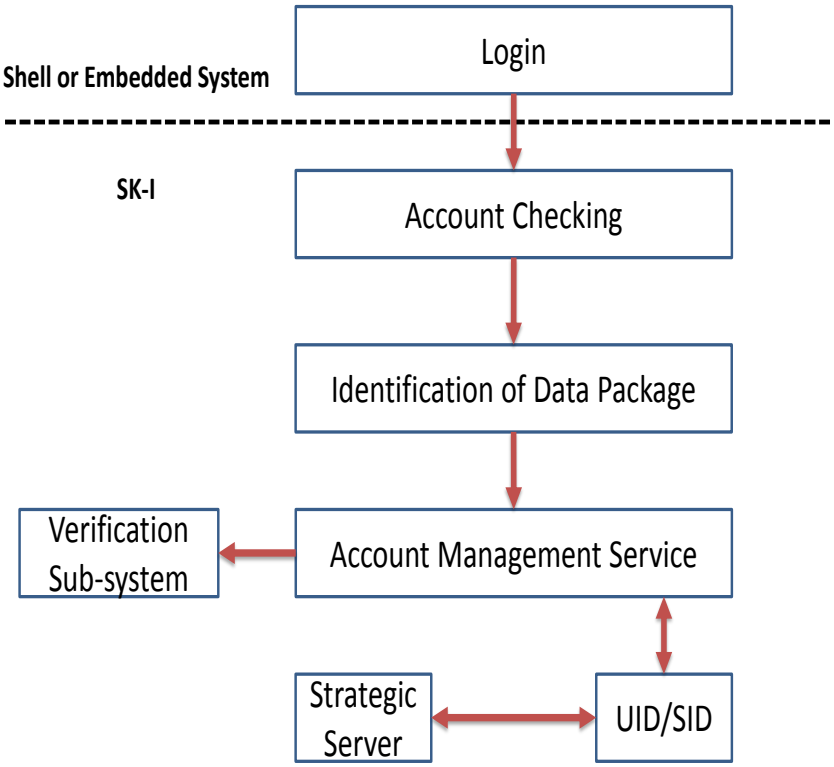

Figure 2. Logics of Authority Service

3.2.7. Strategic Language and Tools: It is used for defining the multi-strategies. The tools management service includes strategy convertor, configuration, and update service, which are assembled as policy manager. These tools run in the embedded system, which are not easy to understand. Over these tools, it is easy to catch by the end-users due to the language compile and definition [10].

In policy manager, strategy convertor works as a compile machine. The user configured strategic text file could be converted into two modes by it [11]. One mode is the .o binary file, which is used for internal loading. The other one is binary file, which is uploaded to the strategic database when SK-I uploads the program. The previous one does not need the support from document system, which suits the simple embedded system.

4. Testing and Discussions

The testing of SK-I includes the master sever and targeted machine. Their configurations are as follows: CPU: Intel ® Celeron ™ 800MHz; Chipset: Intel Corporation 82815/EM/EP/P815/EM/EP; Architecture: AGP x1, PCI x3; Cache: On the CPU card; Memory: 256 MB SDRAM; BIOS: Award Software International, Inc. 6.00 PG; Onboard I/O: Serial Port x2, FDD x2, Ultra DMA 33/66, IDE x2 and USB x2; Hard Disk: WestSpeed 6L040J2, 7200rpm, Buffer 2MB. as well as : CPU: ALIM6117C(i386sx 40MHz); Display: Chips 65545 512K; RAM: 3712K; Real-Time Clock: M48T86PCI; EPROM: 29EE010 120-4C-NH; Ethernet Controller: DM9008F.

4.1. Testing of Functionality

The functionality testing includes several dimensions such as the security label, compulsory access control, user identification, verification, and security control. First of all, the security label is tested through the output information. The output information is

for IPC operation. In this testing, there are 49 operations related to the system security. Thus, SK-I has 2^{49} levels for security labels.

Secondly, the access control uses the security label. SK-I uses four levels for this purpose. The standard is 2 in an embedded system. The user ID is mapped as sid which could be realized over the access control by using the shell.

Finally, the SK-I functionality testing is shown in Table 1, which shows the security items realized in the system.

Table 1. Statistics of Functionalities in SK-I

	Security Label	Compulsory Access Control	User Identification	Verification
Task	Yes	Yes	×	Yes
Storage Domain	Yes	Yes	×	Yes
Memory	Yes	Yes	×	Yes
Message Queue	Yes	Yes	×	Yes
Signal	Yes	Yes	×	Yes
Timing	Yes	Yes	×	Yes
Shell	Uid/sid	Yes	Yes	Yes

Table 1 shows the statistics of functions realized in SK-I. Four key functionalities are tested under 7 items. 'yes' implies the design and realization of function in SK-I, and '×' means the lack of such function. It can be seen that most of the functionalities are achieved in the SK-I for an embedded system.

4.2. Performance Analysis

SK-I performs the security control under four types of strategies: MLS, DTE, RBAC and ACL. The security tasks are divided into 10 levels. In this performance analysis, we use 100 times of function call which spends certain times on security testing and control during the strategy sever. The following Table 2 and 3 illustrate the average time spent on the calling function and decision buffer utilization.

Table 2. Average Time Spent on Calling Functions

Time Unit (ms)	Without SK-I	With SK-I	Increased Spend
Crtos_task_create	694	840	146
Crtos_task_start	210	268	58
Crtos_task_suspend	138	208	73
Crtos_task_restart	145	165	20
Crtos_task_set_priority	288	308	20
Crtos_task_set_note	234	300	66
Crtos_task_mode	32	47	15
Crtos_task_delete	355	489	134

Table 3. Decision Buffer Utilization during Function Call

Time Unit (ms)	Without Decision Buffer	With Decision Buffer	Increased Spend
Crtos_task_create	94	140	46
Crtos_task_start	21	68	47
Crtos_task_suspend	38	103	65

Crtos_task_restart	15	65	50
Crtos_task_set_priority	28	98	70
Crtos_task_set_note	24	83	59
Crtos_task_mode	42	98	56
Crtos_task_delete	95	149	54

From table 2, the results from the comparison of SK-I functionalities report the increased spend which may come from various operations. It can be observed that create and delete operation costs a lot with 146 and 134 unit of time respectively. While, task mode cost the lowest with only 15 unit of time. From the test results, the average cost is 66.5ms which could be accepted by an embedded system for security control.

From the Table 3, the results from comparing the usage of decision buffer in the SK-I. It could be found that the decision buffer plays an important role in enhancing the performance of an embedded system. The testing defines 8 security levels with different operations. For creating operation, the cost is only 46 ms, which is greatly reduced since the decision buffer is able to get the target directly. To this end, Hash table is used in SK-I and this table could be configured according to difference applications so that the decision results could be kept in the table when carrying out various calculations. However, the set priority operation cost with 70 ms is the highest item. That because the priority setting involves all the records in the table so that the update may affect a lot during the operation. It is significant observed that the average cost using the decision buffer is 55.9 ms. That implies the high accuracy of finding the targeted items in the decision buffer. As a result, the performance of SK-I will be improved.

After the testing, the verification of security spends certain level of time, which is constant. For example, when calling different functionalities, the difference of time cost is around 0.5 ms. That means the time cost of such verification could be determined. Thus, an embedded system is able to estimate the time when carrying out the verification operations.

5. Conclusion and Remarks

This paper introduces a security Kernel in an embedded system in terms of design and development of the key components and data constructions. The key components are designed to deal with the challenges when carrying out the security control so as to assist the embedded system. The design considerations are illustrated in this paper so that the lessons and insights from the implementation procedure could be referenced by the practitioners when carrying out the system. Additionally, the data constructions are detailed reported so that each functions and presentations used in the embedded system could be well concerned during the security control.

An example of SK-I which is a security Kernel realized has been proposed to demonstrate the design and consideration. SK-I has been tested under difference dimensions so that the functionalities and performance are examined. From the testing, it is observed that create and delete operation costs a lot and the decision buffer plays an important role in enhancing the performance of an embedded system.

Future research is needed. Firstly, the key components with suitable functionalities and continuous improvements should be carried out so that this SK-I may be used wider in range and applicable in depth. That is very critical for an embedded system that both vertical and horizontal aspects could fit the usage. Secondly, the data construction of different services should be extended since configurable and reusable components and data fields are important during the embedded system security design and implementation. That could reduce the development cycle. Finally, the SK-I could be tested with wider environment so that more lessons and insights could be obtained. Such information will be used for further decision makings like security authorization and verification.

References

[1] N. Qamar, Y. Ledru and A. Idani, "Validation of security-design models using Z", Formal Methods and Software Engineering, **(2011)**, pp. 259-274.
[2] H. Löhr, "Modeling trusted computing support in a protection profile for high assurance security kernels", Trusted Computing, **(2009)**, pp. 45-62.
[3] A. Dmitrienko, "Securing the access to electronic health records on mobile phones, in Biomedical Engineering Systems and Technologies", Springer, **(2013)**, pp. 365-379.
[4] R. J. Richards, "Modeling and security analysis of a commercial real-time operating system kernel", Design and Verification of Microprocessor Systems for High-Assurance Applications, **(2010)**, pp. 301-322.
[5] R. Y. Zhong, "RFID-enabled Real-time Manufacturing Execution System for Mass-customization Production", Robotics and Computer-Integrated Manufacturing, vol. 29, no. 2, **(2013)**, pp. 283-292.
[6] B. R. Kumar, S. Saranraj and S. Tharani, "Enhancing File Security by Integrating Steganography Technique in Linux Kernel", International Journal of Computer Applications, vol. 1, **(2010)**, pp. 70-74.
[7] G. C. Sirakoulis and I. G. Karafyllidis, "Cooperation in a Power-Aware Embedded-System Changing Environment: Public Goods Games With Variable Multiplication Factors Systems", Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on, vol. 42, no. 3, **(2012)**, pp. 596-603.
[8] R. Y. Zhong, "Design and Implementation of DMES Based on RFID", 2nd International Conference on Anti-counterfeiting, Security and Identification, vol. 20-23, **(2008)**, pp. 475-477.
[9] P. M. Özçelik, "A template-based approach for real-time speed-limit-sign recognition on an embedded system using GPU computing, in Pattern Recognition", **(2010)**, pp. 162-171.
[10] A. Elsts, A. Mednis and L. Selavo, "Bayesian Network Approach to Vehicle Mode Monitoring Using Embedded System with 3-axis Accelerometer", International Journal of Imaging & Robotics, vol. 12, no. 1, **(2014)**, pp. 67-80.
[11] M. Winzker and A. Schwandt, "Teaching embedded system concepts for technological literacy", Education, IEEE Transactions on, vol. 54, no. 2, **(2011)**, pp. 210-215.

Author

Liu Shian, He received bachelor's degree in electrical and industrial automation from Harbin Institute of Technology (HIT) in 1996, master's degree in Signal and Information Processing degree from Guangdong University of Technology (GDUT) in 2004. He joined the faculty of Qishuyan Locomotive & Rolling Stock Works under China South Locomotive & Rolling Stock Industry (Group) Corporation in Changzhou, Jiangsu province in 1996. Presently he is working as a teacher, Department of Computer Science and Information Engineering, Guangzhou Maritime College. His current research interests are wireless communication and mobile internet, RFID, computer networks and communication. He has published 20+ papers in various Journals and Conferences of International and domestic repute.

