PhoneProtector: Protecting User Privacy on the Android-Based Mobile Platform

Weizhe Zhang,¹ Hui He,¹ Qizhen Zhang,¹ and Tai-hoon Kim²

¹ School of Computer Science and Technology, Harbin Institute of Technology, Box 320, Harbin 150001, China
² School of Computer and Information Science, University of Tasmania, Virginia Court, Sandy Bay, Hobart, TAS 7001, Australia

Correspondence should be addressed to Weizhe Zhang; wzhang@hit.edu.cn

Received 8 December 2013; Accepted 25 December 2013; Published 4 February 2014

Academic Editor: Sabah Mohammed

Copyright © 2014 Weizhe Zhang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

With the popularity of Android platform based mobile phones, privacy protection of Android platform becomes a focus area. Now protection for Android based smart phone has many shortages, and also most phone protection systems are based on C/S model. In this paper, we propose a browser-free multilevel smart phone privacy protection system, which is based on the Android sensor platform. In this system, protection is ensured by means of SMS, which turns out to be easy, quick, and convenient. Users can send SMS to phones remotely as operating instructions; then the sensors on remote phones execute the instructions and return useful information. Second, the sensors based on the daemon process mechanism are used to prevent the sensors from being maliciously closed and uninstalled. Third, our system adopts SIM detecting mechanism to judge whether the SIM card is removed or changed. If exception is detected, the phone will be locked automatically by its inside sensors. The three points ensure full protection of phone privacy. Test results show that our system has good robustness and low resource consumption.

1. Introduction

Since 2008 when Google first released Android platform 1.0, Android has quick replaced Symbian and plays almost the same important role as iPhone MACOS platform. Android is an open-source, programmable software framework. Users use their smart phones to download free Android applications, which bring much fun and convenience to their life. Convenience results in the fact that more people tend to store their privacy information in their phones. However, privacy information means a lot to the users, compared with how much the phone costs. If privacy information is used by malicious attackers, such as contact information, SMS, and data on storage card, the consequence would be unpredictable. Current privacy protection methods for Android have many shortages: most security mechanisms (screensaver lock, electronic password lock) sometimes prevent users from using phones smoothly. Users often close these protection applications for convenience, in which their privacy would be revealed when phones are lost. Besides, most phone protection systems are based on client-server (C/S) model, which also brings much convenience.

Aiming at problems mentioned above, this paper proposes a browser-free multilevel smart phone privacy protection system, which is based on the Android sensor framework. In this system, operating instructions are sent through SMS to the sensors of remote phones. After quick execution on phones, useful information is returned to users. When the phone is lost, the user can use trusted phone numbers to lock the lost phone. Second, the sensors based on the daemon process mechanism are used to prevent the system from being maliciously closed and uninstalled, which increases the robustness of our system. Third, our sensor framework adopts SIM card detection mechanism to judge whether the SIM card is removed or changed. If exception is detected, the phone will be locked automatically by its built-in sensors.

The rest of the paper is organized as follows. Section 2 discusses background and related works. Section 3 describes the system architecture and design based on the Android sensor framework. Main realization schema is given in
Section 4. Experiment results and analysis are presented in Section 5. Section 6 concludes the paper.

2. Background

2.1. Introduction for Android Sensor Framework. Android is an application execution platform for mobile devices comprised out of an operating system, application framework, and core applications. The Android software stack is built on the Linux kernel, which is used for its device drivers, memory management, process management, and networking. The next level up contains the Android native libraries. Various system components in the upper layers use these libraries, which are written in C/C++. Incorporating these libraries in Android applications is achieved via Java native interfaces. The next level is the Android runtime, comprising the Dalvik virtual machine and the core libraries. Dalvik runs.dex (Dalvik-executable) files are designed to be more compact and memory efficient than Java class files. The core libraries are written in Java and provide a substantial subset of the Java SE packages as well as some Android-specific libraries. The application framework layer, written fully in Java, includes Google-provided tools as well as proprietary extensions or services.

The Android sensor framework can utilize different types of sensors. The sensors can be classified as hardware-based and software-based ones. Hardware-based sensors are physical components built into handset or mobile devices. They acquire the information by measuring related environmental information directly. Software-based sensors just mimic hardware-based sensors. These sensors, named virtual sensors or synthetic sensors, derive their data from hardware-based sensors. In this paper, we propose software-based sensors to protect the mobile privacy.

2.2. Related Works. Shbatet al. [1] provide a security assessment of the Android framework. Various threats may happen by exploiting vulnerabilities in the Android framework to harm, disable, or abuse confidentiality, availability, or integrity of the following Android assets: private/confidential content stored on the device (pictures, contacts, email, documents, and so on); applications and services (phone and SMS applications, Internet, and email); resources such as battery power, communication, memory, and processing power (CPU); and hardware, including the device itself, external memory cards, the battery, and the camera. Benats et al. [2] also point out two shortcomings of privacy management in Android platform. First, Android’s permissions system allows applications to call each other, and under certain conditions one application can make use of any permission that was granted to another application. Second, there is no support for checking conflicts inside the resulting privacy permissions. Then they extend the privacy-aware role based access control (P-RBAC) [3] model to tackle this. Delac et al. [4] illustrate a permission based security model to address security issues. This model is based on application isolation in a sandbox environment [5]. This means that each application executes in its own environment and is unable to influence or modify execution of other application. In [6], an Android-based prototype that implements five collection techniques (Logcat monitoring, BroadcastReceiver monitoring, EventListener monitoring, collection-based collection, and operation-data-based generation) and a selective encryption function to protect sensitive data in the local database is developed. It encrypts parts of operation data for the balance of performance and security. Ghosh et al. [7] propose a semantic policy based system that reasons over a user’s dynamic context and constrains the information flow among applications by extending Android framework. Their semantic web driven policy allows the modeling of complex user context. Google makes it easy for malicious applications to pose a real threat to users’ privacy due to its open philosophy, so yet another android security extension (YAASE) is proposed in [8]. It provides a fine-grained security mechanism while protecting the user from malicious applications that attempts to leak sensitive information via network access or by privilege through collusion. Mobile systems challenges are proposed for routing scheme and secure data transmission for wireless sensor networks [9–11]. Gil et al. propose multisensor architecture to obtain people context from smartphones [12] and Castillejo et al. introduce an Internet of things approach for managing smart services [13]. Rowaihy proposes location privacy in sensor allocation systems [14]. However, none of these works mentioned above use the simple way of SMS to protect users’ privacy based on the Android sensor framework.

3. System Design

Figure 1 is our system architecture with Android sensor framework. The system consists of front-user interface and background service. The user can operate the system in the front-user interface, such as mode switching, information modification, contact information backup, and regaining password. The three subsystems of the background service are phone locking sensor, SMS processing sensor, and defense sensor (self-protection sensor).

Phone locking sensor is in charge of locking the phone and preventing strangers from using it. SMS processing sensor is to process operations related to SMS, including SMS interception, SMS sending, and SMS content reading. Front-end user interface has data exchange with background server: regaining password is realized through SMS processing sensor by sending SMS to remote lost phone. Defense sensor is responsible for the self-protection of our system and taking measures to response to malicious attack, such as uninstalling, deleting, forcibly shutdown, and wiping out data.

Remote phone can send SMS that contains instructions containing password to interact with the sensors. The instruction consists of locking the phone, contact information backup, and formatting the data of the phone. And during backup of contact information, the sensor sends contact information to mail server through data exchange.
3.1. System Function. The system consists of the following function modules: SMS remote control, phone locking and unlocking, phone position locating, contact information backup, system mode switching, user information setting and revising, password regaining, SIM detection, and system self-protection. Next we will discuss these functions in detail.

3.1.1. SMS Remote Control. When the system is in protection mode, background sensor is activated to check the SMS content. If the SMS content fits into the format that our system has defined, then some operations will be activated: locking the system, getting current location of the phone, contact information backup, and wiping out privacy information (contact information, SMS, and data in the storage card) from the phone. One thing to mention, all the operations above are all realized through SMS.

The format of the SMS operation instruction is [PH A B]. A indicates the password to lock the phone. B indicates the operation code. The operation codes consist of three types: 01, 02, and 03. 01 means to lock the phone; 02 means to backup contact information; 03 means to format privacy information. Code 03 could only be sent by trusted contacts; otherwise it is converted to code 01. Operation codes do not belong to these three types which are converted to 01 automatically. Once the remote phone receives the operation code, it would omit operation codes sent from others.

3.1.2. Phone Locking and Unlocking. Users can send SMS using one phone and lock another remote phone. After locking, all other operations except entering password are useless in unlocking the phone. Therefore, our system is defensive to physical attacks: replacing SIM card and storage card, USB data transmission, and reboot. The operations left are only entering password to unlock the phone, making telephone calls, and answering phones. After entering right password, the phone is automatically unlocked. The system only keeps one trusted contact. If the user forgets what the trusted contact, then he can send SMS to it to get the password.

3.1.3. Phone Position Locating. When the remote phone is locked, the system will send the locking confirm information.
to the trusted information, which includes information about current position of the phone. The position information is presented in plane coordinates: \((X, Y)\).

3.1.4. Contact Information Backup. Contact information is critical to the user when the phone is lost. Users want to back up the contact information despite whether the phone is lost or not. It is quite normal if a user can backup contact information during normal situation. By installing our system, we can also backup the contact information through SMS.

3.1.5. Phone Data Formatting. It means to clean up all the data stored in the phone: contact information, call records, SMS, and data in the storage card. It is irreversible.

3.1.6. System Mode Switching. The system switches between normal mode and protection mode. In protection mode, the system starts monitoring sensor in the background to start self-protection function. What we say self-protection function is that the system is forbidden to uninstall and data is forbidden to delete. When received SMS containing operation instructions, it will execute it. However, things are different in normal mode. The system is allowed to uninstall in this mode.

3.1.7. User Information Setting. The system relies on information closely related to the user, such as locking password, unlocking password, trusted contacts, and e-mail (introduced in the following section). Every time the system runs, it checks whether the information is complete. If not, it jumps to the information setting interface. This process is repeated until information is complete.

3.1.8. SIM Card Detection. Each time at start-up, the system will detect the SIM card information automatically. If it changes, the phone will be locked automatically.

3.1.9. System Self-Protection. In protection mode, the system is forbidden to uninstall and data is forbidden to delete. It increases the robustness of our system.

3.2. System Data. The system mainly consists of four kinds of data; see the following discussion.

(1) Locking Password. This is used to lock the remote phone and sent by trusted contact through SMS.

(2) Unlocking Password. This is used to unlock the system when the phone is locked.

(3) Trusted Contacts. When forgetting the password, the user can send SMS to the trusted contact to regain it. Also, 03 operation code is valid only for the trusted contacts.

(4) E-Mail. Backup contact information is achieved by sending all the contact information to the user's e-mail. Also, the user can regain password using e-mail.

3.3. Relations of System Functions and System Data. Figure 2 is the relations of system functions and system data. There are two kinds of data. One is real-time data. The other is global system data, which is the static data of the system.

3.4. System Workflow. Figure 3 is the workflow chart of the main sensor of the system. When the user starts the system, the welcome screen is firstly displayed; meanwhile the system checks the integrity of the user information, which consists of locking and unlocking password, trusted contacts, and email. If the information is not complete, it will jump to the information setting interface and prompt the user to input the complete user information. Then it comes to the main interface; meanwhile the system decides which mode it is in. The display mode is based on this: if in protection mode, the system refreshes the display of the main interface and starts the background service to start the protection module. The system waits for the operation of the user and executes corresponding functions until the user sends exit operation.

4. System Implementation

4.1. System Sensors. The system contains the following sensors.

(1) Front-End Interface sensor. It is responsible for the interaction between the user and the system.

(2) SMS Handling Sensor. It is responsible for handling operation related to SMS: SMS interception, SMS sending, and SMS content reading.

(3) Phone Locking Sensor. It is responsible for locking the phone. The system will be locked forever unless correct password is entered.

(4) Self-Protection Sensor. It is responsible for all kinds of operations that maliciously destroy the normal working of the phone: uninstalling, deleting the data.

(5) Function Mode. It is responsible for executing such functions: mode switching and setting, user information modification, contact information backup, regaining the password, and phone position locating.

4.2. SMS Handling Sensor. The main function of this mode is to lock the remote phone through SMS. It consists of three parts. (1) SMS interception sensor: if the SMS fits into some predefined formats and the password is correct, then some operations will be performed. (2) SMS content reading sensor: the system calls this module to read the content of SMS, just to find if it fits into some predefined formats. (3) SMS sending sensor: it is responsible for organizing some data into SMS and sending it to the remote phone. Figure 4 shows the relations of the three modules.

The implementation of this sensor is discussed as follows. SMS handling module is vital in the system. Even if our system is in normal mode, SMS monitoring is still working. The flow of SMS handling sensor is described as follows. First, SMS comes. Second, the sensor broadcasts the SMS.
Third, the sensor reads the content of the SMS to judge if it fits into some predefined formats. If true, then the sensor checks whether the password in the SMS is right or not. If the password is right, the sensor will intercept the SMS and store the phone number which sends the SMS and also the operation code. Then it will call other sensors to execute corresponding operations. If the password is wrong, the sensor omits it and releases it to the SMS inbox.  

4.2.1. SMS Monitoring Process. As for Android platform, when SMS is received, the sensor will broadcast a message called telephony. RECEIVED to confirm that the SMS is received. Therefore, a broadcast receiver can be implemented to receive the broadcast message SMS_RECEIVED, thus to fetch a message confirming that SMS arrives. However, many applications related to SMS have the function of monitoring the arrival of SMS. Therefore, as for our system, the priority level of monitoring SMS must be higher to all other applications.  

When a SMS arrives, our system reads the head information. If it fits to some predefined formats (see Figure 5), then SMS content reading module is called to read the content of SMS to store the phone number, locking password, and operation code. If the password is right, the SMS is intercepted by our sensor and is invisible to other applications. It looks like the SMS disappears. If the password is not right, the SMS is passed to the inbox.

The sensor will call different modules by different operation codes. The corresponding relations of operation codes and operations are shown in Table 1. Operation code 01 is the default operation code of the system. Each operation code corresponds to a system operation and system function sensors.

---

**Figure 2:** Relations of sensor functions and system data.

**Table 1:** Correspondences of operation codes and function sensors.

<table>
<thead>
<tr>
<th>Operation code</th>
<th>Corresponding operations</th>
<th>Corresponding function sensors</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Locking the phone</td>
<td>Phone locking + phone position locating</td>
</tr>
<tr>
<td>02</td>
<td>Locking the phone + contact information backup</td>
<td>Phone locking + phone position locating + contact information backup</td>
</tr>
<tr>
<td>03</td>
<td>Locking the phone + formatting phone data</td>
<td>Phone locking + phone position locating + phone data formatting</td>
</tr>
</tbody>
</table>
When the corresponding function call is finished, SMS handling sensor will send operation results to the phone number that sends the SMS. The correspondences of system function sensors and returning results are shown in Table 2.

### 4.2.2. Design of SMS Content Reading Sensor

First it reads the content of the SMS into memory. (Sometimes the whole SMS is sent through different parts, so we should wait until all the SMS content is complete.) Then it extracts the part of the SMS based on the format defined in Figure 6.
4.2.3. Design of SMS Interception Sensor. When the format of the SMS fits into the predefined formats and the locking password is right, the SMS is intercepted. This is implemented by terminating broadcast. By this way, our applications are not aware of the arrival of the SMS. This is based on the hierarchical mechanism of Android: broadcast receivers that have higher priorities receive broadcast messages earlier than those who have lower. See Figure 7.

4.3. Phone Locking Sensor. When in protection mode and received SMS that fits into the predefined format, the phone would be locked. The phone locking module mainly consists of four parts: locking the phone keyboard, showing the feedback information of the user, providing contact information of the phone user, and unlocking the phone.

The design scheme of the phone locking module is shown as follows. It mainly consists of two aspects: phone locking condition, phone unlocking condition.

4.3.1. Phone Locking Condition. The condition of locking the phone is satisfied in one of the following conditions: (1) at start-up, when the condition of locking the phone is satisfied; (2) when the sensor received some SMS that satisfied the predefined format to lock the phone; (3) when the system detects the exception that the SIM card is removed or replaced.

4.3.2. Phone Unlocking Condition. The condition of unlocking the phone is satisfied in one of the following conditions: (1) when the user enters correct locking password; (2) when the trusted contact sends correct SMS used for unlocking the phone.

4.4. Self-Protection Sensor. All current software for phone protection can be maliciously terminated or uninstalled. Then the protection loses its efficiency. In response to this, our sensor adopts the communication mechanism of two processes to protect one another at any time. The two sensors are called PhoneProtector and PhoneProtectorHelper. PhoneProtector is first installed during the installation of the system. After that, we will remind you to install PhoneProtectorHelper to further protect the phone. In fact, the two processes are identical. In protection mode, the two processes communicate with each other to confirm the state of one another. If some exception happens, corresponding modules are executed to handle this. In normal mode, the two do not communicate. It is the user's choice to decide whether to fully protect the system or not. However, we strongly recommend users to switch into protection mode.

Now we discuss the communication mechanism of the two processes in detail, as follows.

In protection mode, PhoneProtector and PhoneProtectorHelper detects the real-time state of one another. If PhoneProtectorHelper detects that PhoneProtector is maliciously terminated, it will restart PhoneProtector dynamically. The same way is also done by PhoneProtector. To be specific, we use the broadcast mechanism of Android to achieve the real-time detection of the two processes and take measures when exception happens. PhoneProtector broadcasts its message during fixed time interval in order to let PhoneProtectorHelper know its real-time state. In Android platform, when a program is uninstalled, it will broadcast the message of Intent. ACTION_PACKAGE_REMOVED to all applications. PhoneProtector has registered the message before, so PhoneProtector will response in its receiver module.

In our system, the two processes both start up a detection thread, whose task is to send the message proving that it is still alive to its identical thread. Every 1000 ms, PhoneProtector sends message to PhoneProtectorHelper to prove that it is still alive. We set the time interval to 1000 ms in consideration of both the time interval of broadcast message and the user's response time. Scheduling of threads is realized by the operating system, which cannot be predicted, so too short time interval is not feasible. However, too long time interval is dangerous because during the time malicious attacker can kill the process PhoneProtector quickly and then terminate the whole system.

Our system adopts another mechanism to make sure that the identical process is dead. If PhoneProtectorHelper does not receive the broadcast message of its identical process in 1000 ms, it broadcasts a message to PhoneProtector and record current time. Then in 1500 ms if PhoneProtectorHelper does not receive reply for the message, we believe PhoneProtector has been dead and restart it. The state transition diagram is shown in Figure 8.
To be specific, based on Figure 8, the implementation procedure of the communication mechanism of the two sensors is shown as follows.

1. At first, each process registers the broadcast message of one another. Assume that the messages of PhoneProtector and PhoneProtectorHelper are TELL_PP_IS_LIVE and TELL_PPH_IS_LIVE.

2. The two processes start up their threads and broadcast the corresponding message, respectively.

3. When PhoneProtectorHelper receives TELL_PP_IS_LIVE, it executes operations in corresponding callback function and also records current time. PhoneProtector also does the same thing.

4. When the corresponding thread is called by the operating system, if it does not receive the broadcast message in 1000 ms, then it will broadcast a message to its identical process to confirm if it really dies. If it does not receive reply in 1500 ms, then it labels its identical process to be dead and restarts it immediately.

### 5. Experiments and Analysis

The system is tested in the Testin mobile APP cloud test for 17 types of smart phones. They are HTC A310e, HTC A8180, HTC A9191 4G, HTC ADr6300, HTC Desire G7, HTC Wildfire A3380, Motorola A953, Motorola Droid2 Global, Motorola XT319, Samsung I9003, Samsung I9008L, Samsung I9020, Samsung I9023, Samsung I909, Samsung P1000, Samsung S5570, and MIUI. The test results are as follows.

#### 5.1. Compatibility Testing

We check the compatibility of our system by installing it on 17 different kinds of smart phones. The compatibility passing rate is 100%.

#### 5.2. Performance Testing

We test the starting time consumption, CPU utilization, and memory utilization. The average start time is 0.63 s; the average CPU utilization is 2.9%; the average memory utilization is 7.12 MB. Compared with other phone applications, this resource occupation is relatively much lower. Tables 3, 4, 5, and 6 show the testing results in detail, as follows.

Table 3 shows that the system has low start time, CPU utilization, and memory utilization in average. In Table 4, the start time mainly lies in the region of 0.24–1.02, and the percentage is 76.5% plus 11.8%, which clearly shows that the start time is very fast overall. Table 5 shows that the CPU utilization mainly lies in the region of 1.7–2.4, 3.3–4.0, and 4.1–4.9, which demonstrates that the CPU utilization is low overall. In Table 6, we see that the memory utilization distribution lies mainly in the region of 4.36–8.77. This shows that our system also has low memory utilization.

### 6. Conclusion

In this paper, we propose a smart phone privacy protection system, which is based on the Android sensor framework. In this system, operation instructions are sent through SMS to control the remote phone. When the phone is lost, the user can use the trusted contact to send SMS to lock it. Also, the sensors with daemon process mechanism prevent the system from being maliciously closed or uninstalled. This increases the robustness of the system. Finally, the system adopts SIM detection sensor to prevent malicious replacement or removal of SIM card. Test results show that our system has good robustness and low resource consumption. However, something more can be done based on our works. We use process communication mechanism to prevent the system...
Table 4: Start time distribution of 17 types of smart phones.

<table>
<thead>
<tr>
<th>Distribution region</th>
<th>Start time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.24–0.63</td>
<td></td>
</tr>
<tr>
<td>0.64–1.02</td>
<td></td>
</tr>
<tr>
<td>1.03–1.41</td>
<td></td>
</tr>
<tr>
<td>1.42–1.80</td>
<td></td>
</tr>
<tr>
<td>1.81–2.20</td>
<td></td>
</tr>
<tr>
<td>Numbers in this region</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Percentage</td>
<td></td>
</tr>
<tr>
<td>76.5%</td>
<td>11.8%</td>
</tr>
<tr>
<td>5.9%</td>
<td>0.0%</td>
</tr>
<tr>
<td>5.9%</td>
<td></td>
</tr>
</tbody>
</table>

Table 5: CPU utilization distribution of 17 types of smart phones.

<table>
<thead>
<tr>
<th>Distribution region</th>
<th>CPU utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.8–1.6</td>
<td>2.5–3.2</td>
</tr>
<tr>
<td>1.7–2.4</td>
<td>3.3–4.0</td>
</tr>
<tr>
<td>2.5–3.2</td>
<td>4.1–4.9</td>
</tr>
<tr>
<td>Numbers in this region</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Percentage</td>
<td></td>
</tr>
<tr>
<td>17.6%</td>
<td>29.4%</td>
</tr>
<tr>
<td>5.9%</td>
<td>23.5%</td>
</tr>
<tr>
<td>23.5%</td>
<td></td>
</tr>
</tbody>
</table>

Table 6: Memory utilization distribution of 17 types of smart phones.

<table>
<thead>
<tr>
<th>Distribution region</th>
<th>Memory utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.36–5.83</td>
<td>7.31–8.77</td>
</tr>
<tr>
<td>5.84–7.30</td>
<td>8.78–10.24</td>
</tr>
<tr>
<td>7.31–8.77</td>
<td>10.25–11.72</td>
</tr>
<tr>
<td>Numbers in this region</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>Percentage</td>
<td></td>
</tr>
<tr>
<td>29.4%</td>
<td>35.3%</td>
</tr>
<tr>
<td>23.5%</td>
<td>0.0%</td>
</tr>
<tr>
<td>11.8%</td>
<td></td>
</tr>
</tbody>
</table>

from being uninstalled or terminated. However, if someone is developing a kernel code, he can set the PhoneProtector to be the system process so that it cannot be terminated.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This research was supported in part by National Natural Science Foundation of China (NSFC) under Grant no. 61173145, the National Grand Basic Research Program (973 Program) of China under Grant no. 2011CB302605, and the National High Technology Research and Development Program of China under Grant no. 2011AA010705.

References


