Secure Resource Sharing in Ad hoc Networks

Michaela Greiler, **Peter Schartner**
System Security, University Klagenfurt, Klagenfurt, Austria

Abstract *Mobile and wireless devices become more popular and their capabilities increase as well. At the same time, most of the today's PC power is unused, as a study of the Gartner Group indicates. Grid computing and P2P systems take advantage of this fact and provide unused resources to other nodes within the network, but mobile scenarios or ad hoc networks are often left out. This paper presents a concept for secure resource sharing in mobile ad hoc networks, with a special focus on challenges and problems with mostly unknown and potentially unreliable and untrustworthy devices. Virtualization technologies will be presented as a solution to grant application security, as they are also used in grid and trusted computing. The secure resource sharing (SRS) system, that will be presented in this paper as well, focuses on granting security aspects like availability, privacy, confidentiality and trust establishment in the area of resource sharing.*

Keywords: secure resource sharing, security aspects, virtualization, sandboxing, authentication, trust.

1 Introduction

Today everyone talks about local services. Separate applications provide different services like information about restaurants, traffic congestion, and tourist information. Normally all these services are obtained and managed from a central server and provide and distribute information and data. Nowadays mobility is one of the most pressing topics, and small, mobile devices become more popular every day. Problems with these small devices like cellular phones, personal digital assistants (PDAs) and small laptops are limited central processing unit (CPU) power, small disk space, little available random access memory (RAM) and limited communication bandwidth. Furthermore many applications are not executable because of hardware restrictions or incompatible operating systems.
Gartner Group research results in the perception that over 95% of today's PC power is unused [5]. Many different communication technologies are available, but cooperation abilities between them are constrained. For example, every type of network connection (e.g. Infrared – IrDA, Bluetooth, LAN, WLAN) within a Windows operating system has to be configured and updated separately. If someone wants to find communication partners in range, every connection type requires a request or an update. Moreover, communication is only possible if the hardware of both communication partners is supporting the desired network connection. The main idea of the secure resource sharing (SRS) system is, to go one step up and abstract from different communication technologies to provide a technology independent communication and service infrastructure. Services should provide resources to small and restricted devices. These services not only include information and data services, but also provide CPU power, applications, and disk space.
At this, the main focus is set on the security aspects of resource sharing, and the restrictions that come along with ad hoc networks. Figure 1 shows a potential network in which resources can be shared. In this network, devices of different power and abilities exist. For example the smartphone can only communicate with the other devices via infrared. Only the PDA is able to communicate via infrared and WLAN, and is used as a bridge to other devices like the laptop, which further allows to access the printer or the Internet via the PC. Furthermore, the laptop can request resources from other devices. For example, the laptop may outsource some costly calculations to the more powerful PC.
Every user who wants to participate, has to download the middleware from a central third party (generally a server). The middleware is responsible for network management, which includes opening and closing the network connection as well as mediating between different types of communication technologies. Other tasks like creating and maintaining service lists are covered by the middleware too.
Impartial from the underlying communication technology, a high level protocol (at application level) will be used for service communication. Security tasks like authentication, assurance of integrity etc. are covered by the middleware as well as by the services themselves. A service is independent from the middleware and can be seen as a plug-in. The middleware provides a framework for network management, communication and security. The services or plug-ins extend this framework and can be added ad libitum. This architecture provides a common basis for various services from different software producers. The middleware offers a well defined and open API. A similar approach to split middleware and services is used by the

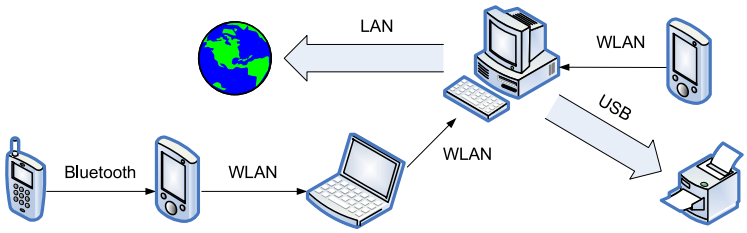

Figure 1: Potential resource sharing network

SMEPP project, that focuses on providing a middleware for P2P systems [8]. The concept of a shared middleware and the ability to integrate various different services should lead to a consistent service environment. By now, services are grouped in four categories:

Information Sharing: Information services, often implemented as mobile agents, offer the ability to drop search queries that return some information collected by other devices in range or from the Internet. Examples are getting weather forecasts, finding petrol stations, restaurants, books etc. Information services can not only be used to obtain information or data, but also to distribute it and transfer it from one point to another implicitly, like it is useful for emergency services. In addition, information services are thought to allow peers to work in a collaborative manner. Therefore white boards, instant messaging, and co-browsing can be realized.

Storage Sharing: Services belonging to this category provide free disk space from devices within range. For example someone wants to outsource data because the local disk space is limited. Another device in range has enough capacity to store the data without any constrictions or obstructions. The service would mediate between these two devices and allow the restricted one to save as many data on the other device as possible and approved. Data integrity and confidentiality can be assured by the use of encryption and data integrity mechanisms. But at one go encryption offers another problem: namely that the part that saves the data does not know which data it is going to save. This is known as data legitimacy problem.

Computational Sharing: Using the computational power of another device is critical and must be well-conceived. The idea of computational sharing is to use a conglomerate of many idling devices in range, to which computational tasks are outsourced. These devices compute the tasks and return the results to the applicant. In addition to transferring the data used for corporate computation, the algorithm itself can be transferred too. To warrant the safeness of the calculating device, virtual machines or processes can be used (sandbox techniques). In that case, the whole computation will take place in a shielded environment.

Application Sharing: This category includes all services offering applications from other devices to be used by applicants. Similar to computational sharing, virtual machines can provide safety for the service provider. An example for such a service is an export application. An export service could allow an applicant to send data, for example a Word document, to the service provider which converts this into a PDF document and sends it back. Besides the high risks, which are caused by invocations of programs, legal and license regulations come into play.

The paper is structured as the following: First the usage of the SRS system at a conference is exemplified. Then virtual machines (VMs) and sandboxes are explained, which are thought to provide application security. Afterwards the use of VMs is motivated for application and computational sharing. Finally pending questions and future work are presented.

2 Conference Scenario

An appropriate scenario for the usage of the SRS system is a scientific conference. Many people with similar interests meet for the same goal - participation and knowledge transfer. To come into contact and get to know each other, a special information service can be used. Let us call it business card service. As its name implies, this service provides business cards. Everybody having installed this service, distributes business cards to other conference participants and in return can receive business cards from the others too, as illustrated in Figure 2 (left).

Furthermore the service can also provide information about contacts of a person, like it is often used in social networking sites (e.g. LinkedIn). Other information sharing services allow participants to work in collaboration, like messaging services, services that provide a shared white-board or enable co-browsing.

In such an environment, business and computer collaboration can be extremely relevant. Authorization can be guaranteed through network access authorization, which in particular means that every conferee has got a username and a password to gain access to the network. Simultaneously, the possession of this access credentials can be used as authorization for service usage.

Using the capabilities and resources of many devices at a conference can provide great potential and the output of this computation assembly can be enhanced. Another potential scenario for the use of the SRS system is sharing a data projector, as Figure 2 (right) shows. Normally only

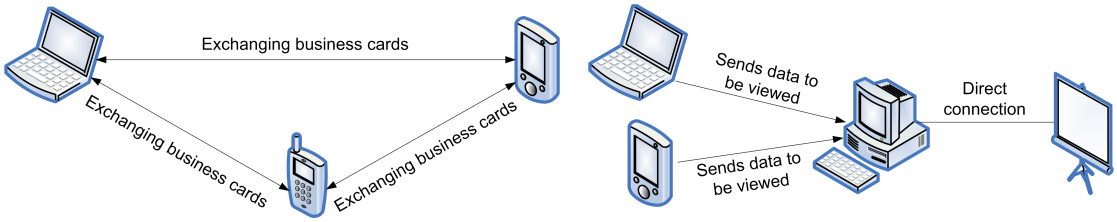

Figure 2: Exchange of business cards and Projecting data on a shared screen

one device, for example near to the speaker's desk, is connected to a data projector. All conferees or speakers that want to provide some data have to carry their own device to the lectern and connect it to the data projector. A further possible solution would be to save the content on a portable hard disk (e.g. USB) and copy it to the connected device. The SRS system can help to simplify this scenario. The content will be delivered (e.g. via streaming) to the connected device. No carrying around of devices is necessary. The only requirement is that both the delivering device and the receiving one share the same application sharing service. Further examples for resource sharing are knowledge sharing services, that allow conferees to access information about research interests of other participants or share research papers, and services that allow access to printers, data projectors and other accessible devices.

Storage sharing at a conference enhances the disk space of devices by using unused disk space of nodes in range. Often conference members do not want to take more than a smartphone or a PDA to a conference. Let us consider a use case where the conference participants have left their laptops at the hotel. Unfortunately, when they arrive at the conference they notice that interesting research papers and prototypes are provided for downloading today. A storage sharing service can be deployed to allow conference members to store their data on different devices.

The processing of statistical tests is a predestined example for outsourcing of data processing and the researchers use the ability of the SRS system. During the conference, a group of researchers presents its collision-free key generation (CFNG) algorithm. The output of the key generator is claimed to be unique and random. To test the randomness of the generated key material, the researchers want the output to be tested by many statistical tests. The more tests the output of the key generator passes, the more it can be deemed to be random. A particular scenario is illustrated in Figure 3. There researcher A sends one algorithm for a special statistical test (e.g. one of the Diehard Statistical Test Suite) referred as suite1 to CPU providers P_1 and P_2. Another test suite, named suite2 that is known as resource-intensive and time-consuming is sent to P_3. Then A distributes to every provider another part of the CFNG output. The providers start the statistical test and after finishing they return their results to researcher A. The results of the test suite2 are not expected until next day. Therefore the researcher and also the provider will disconnect from the network with high probability resulting in an availability problem. The basic idea to solve this situation is that providers having already finished the computation will store the results for a particular period of time (e.g. one day) until researcher A is available again. If A is in range again, they will send the results immediately. By the next day, the researchers can use all the received results to present the randomness of their CFNG to the conferees. Beyond using the SRS system at a conference, other environments and situations, in which the SRS system can be useful, are bus stations, airports, office buildings etc.

3 Virtual Machines & Sandboxes

Besides all efforts to secure communication between nodes and to introduce authentication mechanisms for ad hoc environments, it is crucial to concentrate on application security. To allow applications and code from other, mostly unknown, nodes to be executed, involves the risk that the code that has been downloaded and installed is malicious. To download code only from sources that are fully trusted is not practical in many systems, and claims for full trusted sources trespass the ad hoc systems' character. Furthermore it is not possible to check if the code is malicious or not prior execution. Securing the application execution is important, because willingness of the users to share their resources will reduce drastically if the participation is associated with a serious risk [5], [2]. As usual, programs that will be invoked at the supplier's device are executed with all the privileges of the supplier. This means that the program may have write and read access to the user's disk, may be able to open network connections and create other processes. These privileges have to be restricted for programs invoked during resource sharing. It has to be guaranteed that the state of the real machine of the supplier will not be changed, the privacy of the supplier is protected and that no unwanted programs or processes will be executed. Limitations of privileges and access to resources can help to prevent malicious code from harming the user's system. The downloaded code has to be executed in such a way that it can be completely controlled. Therefore access to computer resources will be restricted according to the trust level of the code. Restricted resources can be local disks (read and/or write permissions),

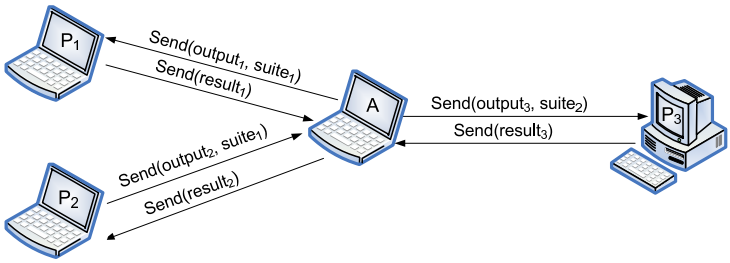

Figure 3: Distributed computation for testing randomness

network connections, process creations, and access control for dynamic libraries or native methods. Virtualization technologies enable to run untrustworthy and unsafe applications in a virtual environment that protect the real environment from permanent damage [6]. Therefore one or multiple execution environments are created on a single physical machine, as Figure 4 illustrates. Each virtual machine is a reproduction of the real physical machine. These virtual machines are distinct, isolated and do not interfere with each other or the underlying physical machine. The virtual environment can be analogue to the current host environment, by copying the environment of the physical machine to the virtual machine.

Different approaches exist to enforce application security by restricting access and privileges. Security Enhanced Linux (SELinux) technology uses Access Control Lists to limit system access. The User Mode Linux (UMLinux) and Virtual Server (VServer) technologies create different virtual servers for execution of unknown applications. UMLinux can be seen as a linux-inside-linux solution. The virtual machines protect the real machine from being damaged, because changes on the virtual machines do not influence the real system. A drawback of this approach is the decreased performance because of running multiple operating systems within one device. With the sandbox security model, a much better performance can be achieved than by creating complete virtual machines. The sandbox security model tries to provide a restricted environment in which untrustworthy code can be run by separating processes and limiting access to the system resources (e.g file system). The Java architecture is well-known for the use of a sandbox model. In Java the sandbox security model uses a security manager to determine the access rights. Code, remote or local, can be subject to security policies. Other examples for sandbox approaches are the chroot command, and jailing in Free-BSD (Berkeley Software Distribution). The chroot command restricts the file system operation for a process. Jail is a combination of the chroot command and the sandbox approach. Many approaches, such as UMLinux and Jail, are restricted to particular operating systems. But in many cases such limitations are not preferred. A heterogenic environment, such as in the SRS system, requires a technique that is independent from the underlying operating system and hardware. Furthermore combinations of the methods listed above have to be considered. The Java 2 Platform, Micro Edition (J2ME) gives a direction to secure applications but in addition focuses on small and powerless devices like smartphones [7]. Key security features of J2ME, assuring application security, are bytecode verification, code signing and sandbox mechanisms. Bytecode verification is an important feature that restricts an application, in such a way, that it can only access memory and resources within its domain. It prevents an application from overloading the Java language core libraries. Code signing is the next major pylon of the security architecture. Digital signatures are used to check the origin and the integrity of the data. Only applications that pass the signature verification process can be downloaded and executed. Furthermore, access permissions of the application depend on the trustworthiness of the application's source. Hence read and write operations on the local storage or establishing a network connection can be forbidden. For network and data security, J2ME allows to use point-to-point secure connections. Protocols such as the Secure Socket Layer (SSL) and the Transport Layer Security (TLS) protocols are supported and the Secure Hypertext Transfer Protocol (HTTPS) is available in the Generic Connection Framework. Data security can be achieved through the Record Management System (RMS), which allows encoding data.

Virtualization is often reviled as heavy-weighted and clumsy, and sometimes for good reason. One of the problems is that frequently VMs are complete operating environments, although only a subsystem is needed. The initialization of virtual machines is an expensive task and leads to costly disk usage and invocation latency.

Software virtualization technology is located between the operating system and the applications. Applications executed in the virtual machine can access all resources of the physical machine but are restricted in such a way, that they cannot tamper the real machine. The virtual machine shares the execution environment with the real machine. Deviations from the host environment are stored in the VM's local state. These deviations, or a subpart of them, can be synchronized with the host machine if desired.

Often the concept of namespaces is used for virtualization and separation. Each VM acts in a distinct namespace and all resources are renamed to the new namespace of the VM

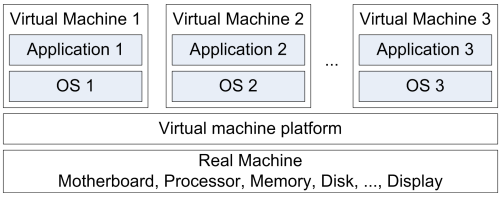

Figure 4: Two VMs running on one single device

they belong to. Namespace virtualization is used in many well-known systems like FreeBSD Jail, Solaris Containers and Linux VServers. Yu et al use namespaces for creating virtual machines for windows applications too, and they call their virtual machine feather-weight virtual machine (FVM).

Other windows-based virtualizations are Progress Deployment System (PDS) and Softgrid. Softgrid erstwhile Softricity provides application virtualization and dynamic streaming technologies [4], [6]. PDS creates separated VMs with a subset of the Windows APIs. Other existing virtualization technologies are VMware and virtual PC. In contrast to Softgrid they virtualize at the hardware abstraction layer. Bochs is an emulator project that runs on most popular platforms and can be found at http://bochs.sourceforge.net/. The complete x86 PC emulator allows running operating systems and software within the emulator and therefore enables the user to have a machine inside a machine. Virtualization technologies can be differentiated by their level of abstraction. Hardware abstraction layer virtualization, like used by VMware, Virtual PC, Denali, Xen, and User Mode Linux abstract the hardware of the physical device, like memory, processor and peripheral I/O devices in such a way that multiple instances of operating systems can be installed on a single machine. A major advantage of hardware layer virtualization is the complete separation and isolation of the different virtual machines and the host machine. Disadvantageous is the overload and bad performance of the technique because of the high resource needs.

At the same time, virtualization can be done at the operating system level. FreeBSD Jail, for example, uses multiple virtual execution environments to separate resources that can be accessed by an application. These virtual environments are called jails. The linux-based VServer separates the user-space environment into distinct virtual private servers. Other virtualization environments for the Linux platform are Sphera and SWsoft's Virtuozzo that can be found at http://www.swsoft.com/en/products/virtuozzo/. In addition to Linux, Virtuozzo supports the Microsoft Windows server operating system for virtualization. Solaris Containers offers virtualization technologies for the Solaris operating system.

Calder et al present in [1] a virtual machine for desktop grids, named Entropia. Entropia is designed to safeguard desktop PCs that offer their resources for grid computing from damage caused by malicious or malformed applications. Additionally, this environment monitors the behaviour of the applications and the usage of the resources to guarantee that the PC user will not be interrupted or disturbed. Access control mechanisms restrict the grid application from modifying data on the device or gaining access to private data of the user. Entropia pledges that the system of the user will be in exactly the same state after the completion of a task as it was before, and that the user will not experience a performance decrease during working.

Calder et al define requirements for a virtual machine used for securing the participation in grid computing, which are:

Integrity of the device: The data of the device should neither be accessible nor writeable for the grid computing application. The integrity and confidentiality of the data has to be guaranteed.

No change of the state: The state of the real machine has to be exactly the same after the execution of the grid application as it has been before.

No performance decrease: The user should always be able to use his or her device unhindered. If different processes draw on the same resource, the grid application processes have to be non aggressive and run with lower priority.

Integrity of the grid computation: Not only the user can claim protection of his or her device and the data, but also the integrity and confidentiality of the grid application and its results have to be considered.

Entropia uses a desktop controller that monitors and controls the entire grid jobs running on the supplier's machine. This controller guarantees no performance decreases for the user. It looks for the disk, the CPU, the input and output and the memory usage and restricts the amount of processes that can be started by the grid application. If the grid application uses the resources too heavily the desktop controller will throttle, pause or even terminate the application. Moreover, Entropia wraps all the grid applications by using binary modification technology. It wraps native binaries which leads to the advantages that the application can be written in any language that compiles to x86 and no source code is required. A detail description of the wrapping process can be found in [1].

Furthermore Entropia offers sandbox mechanisms that mediate the complete grid application interaction with the operating systems and guarantees integrity and confidentiality for the user's data, the grid application and the according data.
As it can be seen, much research has been done and will be done in future on virtualization technologies. The overhead of running virtual machines can be reduced more and more. As Yu et al show, their FVM has less than 20% overhead while execution of command line programs. The potential of virtualization, especially for deployment of mobile or untrustworthy code and applications, is enormous. Also in the SRS system sandboxing and virtualization technologies play a key role. Applications that a user runs in a normal system are allowed to invoke any system calls that the user is allowed to make, because the programs are executed with all the privileges of the user that invokes it. Defining access policies can help to prevent some applications from deleting files, modifying registry entries, making a network connection and so on. The problem of this approach is the creation of the policy files. It is very likely that a "good-natured" application is too restricted and it cannot conclude its work, but on the other hand a malicious application sometimes has too many privileges. As most commonly there is insufficient prior knowledge of access requirements, correctly defining which functions should be blocked is complicated and often not successful.
Virtualization in contrast to sandbox mechanisms offers the ability to execute the untrusted application in an environment that seems to the application as the unrestricted host environment. In general, modifications of the physical machine are not allowed, but can be selectively committed and transferred to the host environment. Transfer of state modifications to the host environment is one of the most challenging tasks. How can desired modifications be distinguished from unwanted ones? This question takes a back seat in the SRS system, because most of the times the requirement for changes to the host environment is not given. In the case of sharing CPU cycles, no permanent modification of the host environment is needed, only the execution of the applicant's desired application has to be done. The only permanent modification of the host system that can be required from SRS services is to store data on the disk. This data can be the results of the computation or naturally the data that should be stored for the applicant. The type of data that has to be stored does not include executable data.
After presenting technologies that are closely related to and useful for the SRS system in this chapter, the next chapter focuses on how these concepts can be applied to application and computational sharing.

4 Application & CPU Sharing

The major problem according to application sharing is the process of invoking programs on another device. How can the supplier be sure that its device will not be damaged? Which programs are allowed to be invoked, and which are not allowed? Invoking programs is a critical task, because often through these programs security holes can be used to, for example, accessing private data of the supplier or even invoking some other programs that should not be accessible to the applicant. We refer to securing the invocation and execution of programs as *application security*. A program that is invoked normally runs with all the privileges of the supplier, which means that the local disk can be accessed for reading and writing, processes can be created and network connections can be opened [3], [1]. Therefore the process of invoking has to be well designed and secured. It must be granted to the supplier that the supplier's device will be protected from unwanted access or data modification, that the supplier's device will not be harmed by malformed or malicious applications, and that the supplier can control the usage and behaviour of the service executed at all times, in such a way that the user will not experience unwanted deterioration of performance.
One important problem besides harming the provider's device is the submission of false results. How can an applicant be sure that the received results are correct? Can an applicant recognize that the results have been tampered? Protecting the *authenticity* and *integrity* of the sharing application can increase the trustworthiness of the results. Therefore the supplier should not be able to manipulate the sharing application and the results.
Virtualization techniques can be used to hinder a provider from sending false results. The main idea is to use an enclosed, secure virtual machine that gets the input data in an encrypted way and provides the results encrypted too. Only during data processing the input and output will be available in plaintext. Because the result is encrypted when it leaves the secure virtual environment, it is difficult for the adversary to manipulate the cipher text in such a way that it will be accepted as a valid result. The complexity of manipulation depends on the applied encryption scheme and additionally the used hash function. This process is illustrated in Figure 5.
In the case that an adversary gets access to the secure virtual machine or the data, and the algorithm is not encrypted, an attacker can easily send false results.
To make it more difficult for an adversary to submit false results the integrity and authenticity of the service application has to be protected too. This means that the service application should not be modified by a supplier and the supplier should not be able to modify and read results and input data. If results have to be stored they should only be stored encrypted on the device. The virtual machine

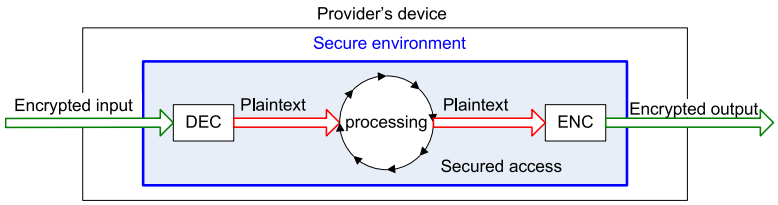

Figure 5: Secure Processing

is responsible for providing a secure environment that is not accessible for the user. Furthermore the integrity of the service application has to be proven before execution. This proof can be based on encrypted checksums of the binary data files of the application. Before execution of the application the checksum is validated and indicates whether an application has been tampered or not. This procedure for example is also used in Entropia [1]. Similar to computational sharing, virtual machines can be used to allow a secure and isolated invocation of programs. The process is as follows:
For all the services, a separate virtual machine will be started, and programs can only be invoked within this virtual environment. To make the virtual machines, the service applications, the algorithms and the data nearly inaccessible a trusted hardware, like a smart card or related to trusted computing the trusted platform module, can be used. This module stores for example the secret keys needed for the encryption of the input and output data, the application etc. The use of smart cards is preferable to the use of a built-in module because cards can be exchanged easily, depending on the current application needs. The virtual machine is furthermore responsible for restricting and maintaining the access to local resources of the supplier's device, like disk space, network connections, and process creation. Also, no permanent changes are allowed to take place. The whole process of application sharing is temporary, and the status of the live system will not be changed. This means that neither the application sharing process will write data to the hard disk or change registry entries, nor should private data of the user be accessible. After completion of the application sharing process, the system of the supplier should be exactly the same as it was before. Therefore the program that will be invoked has to run with restricted privileges. Virtualization techniques, like the ones described before, are a must and can grant beyond application security, confidentiality, and a higher trust in the authenticity of the results.

5 Resumee and Future Work

In this paper we presented a concept for secure resource sharing in ad hoc networks as a first step of a current research topic. Concerning shareable resources, we have identified four resource classes by now: information, storage, CPU-power and applications. Technologies which might be useful to implement this concept include virtualization and sandboxing. The next steps include a detailed analysis of special security needs and technologies and mechanisms like trusted computing or anonymous authentication. Besides technical aspects, there are several legal aspects which have to be investigated, too. Here we will focus on techniques providing traceability of actions, non-repudiation and conservation of evidence, which may be of importance when concerning fears of potential users of the SRS system. After refining and elaborating the concept, we will implement a prototype, in order to perform some performance and security tests.

References

[1] B. Calder, A.A. Chien, J. Wang, and D. Yang. The entropia virtual machine for desktop grids. In VEE 05: Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments, pages 186-196, 2005.

[2] E. Dodonov, J.Q. Sousa, and H.C. Guardia. Gridbox: securing hosts from malicious and greedy applications. In MGC 04: Proceedings of the 2nd workshop on Middleware for grid computing, pages 17-22, 2004.

[3] J.Y. Levy, L. Demailly, J.K. Ousterhout, and B.B. Welch. The safe-tcl security model. In ATEC98: Proceedings of the Annual Technical Conference on USENIX Annual Technical Conference, pages 23-23, 1998.

[4] Microsoft. Microsoft completes acquisition of softricity, last accessed 24.02.2008, 2006. http://www.microsoft.com/presspass/press/2006/jul06/07-17SoftricityPR.mspx.

[5] I. Taylor. From P2P to Web Services and Grids Peers in a Client/Server World. Springer, 2005.

[6] Y. Yu, F. Guo, S. Nanda, L.C. Lam, and T.C. Chiueh. A feather-weight virtual machine for windows applications. In VEE 06: Proceedings of the 2nd international conference on Virtual execution environments, pages 24-34, 2006.

[7] M.J. Yuan and J. Long. Securing wireless j2me - security challenges and solutions for mobile commerce applications. Technical report, IBM, University of Texas, 2002. http://www.ibm.com/developerworks/wireless/library/wi-secj2me.html.

[8] M. Albano, A. Brogi, R. Popescu, M. Diaz, J. Dianes Towards Secure Middleware for Embedded Peer-to-Peer Systems: Objectives & Requirements. In proceedings of the 6th International Workshop on the Foundations of Coordination Languages and Software Architectures, Lisbon, Portugal, 2007.