Privacy Guaranteeing Execution Containers: One time use of personal data by location based services

Peter Langendoerfer
IHP microelectronics
Im Technologiepark 25
15236 Frankfurt(Oder), Germany
+49-335-5625350
langendoerfer@ihp-microelectronics.com

Michael Maaser
IHP microelectronics
Im Technologiepark 25
15236 Frankfurt(Oder), Germany
+49-335-5625
maaser@ihp-microelectronics.com

ABSTRACT
Privacy issues are becoming more and more important especially since the cyber and the real world are converging up to certain extent when using mobile devices. Means that really protect privacy are still missing. The problem is, as soon as a user provides data to a service provider the user looses control over her data. The simple solution is not to provide any data but then a lot of useful services e.g. navigation applications cannot be used. In order to remedy this problem we propose privacy guaranteeing execution containers (PGEC). Basically the concept is that the application gets access to the user data in a specially protected and certified environment, the PGEC. PGECs enable applications to access private user data locally and guarantee that the user data is deleted as soon as the service is quit. Thus, the PGEC guarantees a "one time use" of the provided private data. The PGECs also restrict the communication between the application and the service provider to what is explicitly allowed by the service user. In order to highlight the security provided by the PGEC, we discuss potential attacks such as modified execution environments as well as appropriate countermeasures.

Categories and Subject Descriptors
D.3.4 [**Processors**]: *Run-time environments* D.4.6 [**Security and Protection**]: *Access control, information flow controls*

General Terms
Algorithms, Management, Design, Economics, Reliability, Security, Human Factors,

Keywords
privacy enhancing techniques; P3P; location based services.

MOBILWARE 2008, February 13-15, Innsbruck, Austria

DOI 10.4108/ICST.MOBILWARE2008.2845

1. INTRODUCTION
The Internet provides us with access to latest news and shopping facilities 24 hours a day, 7 days a week. This is really amazing since it simplifies our everyday lives. Third generation mobile devices even add a new dimension, i.e. now we cannot only access all these services at any time but also at any place. In addition this allows tailoring certain services to the users' current contexts, i.e. their current position can be taken into account when searching for restaurants etc. But this all is a serious risk for users' privacy. Especially the integration of mobile devices in those service architectures allows to link real world and cyber world behavior, so that detailed profiles can be gathered.

From our point of view the fundamental problem is that anyone who receives data for whichever purpose has the capability to copy, store and distribute these data. In order to tackle this problem we introduce the concept of privacy guaranteeing execution containers (PGEC). The basic idea is that personal data is not directly exposed to the service provider but accessible inside the container. PGECs allow to combine sensitive user data e.g. current position with sensitive service provider data e.g. navigation algorithms in a secure way. In very simple words the PGEC guarantees a "one time use" of user and service provider data as well as of service providers algorithms. The key components of our privacy guaranteeing execution containers as well as appropriate protection means are prototypically implemented.

The rest of this article is structured as follows. We start with an overview of existing privacy protection approaches. Section 3 provides the description of the container concept. As part of this section we discuss the requirements that result from different kinds of applications. Implementation issues are investigated in section 4. The paper concludes with a summary and an outlook on further research steps.

2. RELATED WORK

2.1 Standards
Technically P3P [1] defines an XML dialect for the description of privacy policies. So service providers can state which data they are gathering for which purpose. APPEL [2] can be used to express what a user expects to find in a privacy policy. P3P and APPEL merely provide a mechanism to describe the intentions of both sides rather than means to protect user data after agreeing to use the service.

IETFs GeoPriv working group is developing an architecture for handling location information in a privacy aware manner [3]. One of the benefits of this architecture is that the privacy rules, are stored as part of the location object [3]. Thus nobody can claim that she did not know, that access to the location information was restricted. But misuse is still possible and it is still not hindered somehow by technical means.

2.2 Location-aware Platforms

There are several approaches that try to protect privacy in location aware middleware platforms [4],[5],[6],[7],[8]. In [4],[5],[6] means are discussed that enable the user to declare how much information she is willing to reveal. [7] discusses a middleware that uses user defined rules, which describe who may access the user's position information and under which circumstances. The approach investigated in [8] intentionally reduces the accuracy of the position information in order to protect privacy. This helps to protect privacy to certain extent, but it cannot be used in systems that need an accurate position to work properly, e.g. navigation services. In all these approaches means to enforce privacy are missing.

2.3 Protection Means

There is a lot of work done in the area of digital rights management to protect content [9],[10] as well as code from misuse [11]. Those approaches rely on specialized hardware such as Smartcards, or are vulnerable to data extraction [12]. Those systems do not provide means to execute any code to be freely defined as it was needed for services. They merely protect media content, which could be considered as the service provider's data. But, despite the protection of user data is in principle an equivalent problem these approaches do not provide a solution for protecting service users' data.

To the best of our knowledge there are only two approaches [18],[19] that try to make sensitive data available to a third party while ensuring secrecy of that data. [19] proposes an architecture that ensures secure data processing by exploiting the java sandbox model as execution environment for data processing code and by limiting the feedback from the data processing code to the out side world. In order to allow correct interpretation of data processing results as well as development of appropriate algorithms a part of the data has to be publicly accessible. In addition sensitive data is always kept at its owner's site. The prerequisites of this concept render it impractical for implementation of location based or context sensitive services, although it is well suited for privacy preserving data mining.

The approach presented in [18],[13] tries to avoid that user data is accessible outside a specially secured execution environment. User data is enclosed in an agent and securely transferred into an isolated closed-door one-way platform provided by a trusted third party. The service agents proceed analogous with their own data. Those entire agents interoperate within that trusted environment and agree on a certain result. The result is forwarded by all involved agents independently to the closed-door platform which posts the result to the agents' origins if the forwarded results are equal. This ensures that no private data is transmitted to the opposite party if the agent did not agree to. All the agents together with their enclosed data are deleted after service completion in order to ensure the privacy of the user and to protect the services data. In contrast to this approach PGEC does not rely on a trusted third party that provides processing capabilities such as a server plus a specific agent platform. Encapsulation of sensitive data and its deletion after service completion are provided by the PGEC by design. Thus it allows for bilateral cooperation between service users and service providers. User and service provider data do not need to be transferred a priori, but only when really needed or may even be used without being transferred via the network. This is especially helpful if location based services are realized inside the container since they may need a huge amount of data such as a catalog of all restaurants in New York City. This feature is ensured by the concept of a distributed PGEC which consists of at least 2 instances at either participant side, and which transparently represents virtually one single PGEC.

2.4 Regular Firewalls

A number of firewall systems exist in the market. Those are either software solutions so called personal firewalls such as the MS Windows Firewall or hardware solutions built into routers, i.e. using Linux' iptables. Mostly those are used to prevent unwanted access from the outside of the firewalled hosts. Firewalls can further be used to block certain protocols or several ports. As far as it is known there is no way of a firewall to prevent certain data items to be sent to not blocked addresses. Even though there is some packet inspection there is no chance to recognize the content of encrypted packets. Firewalls have no control over the data once it passed the firewall. Hence, a firewall may only assure that no data leaves the machine at all. This is similar to unplugging the network cable which renders all internet services unusable.

In opposite to the firewalls the PGEC is supposed to transfer sensitive data securely over the network into other PGEC instances only. While regular firewalls do not prevent local applications to write onto local hard drives or similar persistent storage devices - even printers, the PGEC does. That way all data transferred within PGEC instances remain within those. This is true for private data of service users as well as data and service code of the providers. Within the container instances both can be brought together and the service can usefully be completed. Since data can only be used within PGEC instances this principle may be compared to DRM systems where the protected content can only be decrypted and played within certified DRM enabled systems, which comply with the requirement not to persistently store the decrypted data.

3. CONTAINER CONCEPT

In order to provide useful context aware services a lot of different information has to be taken into account. The range spans from personal profile of the service user over related data from a third party to the algorithms used and developed by the service provider. The major problem to solve when it comes to privacy issues is how to guarantee that the service provider does not retrieve data of the service user. This task has to be tackled in such a way that the solution can adapt to different contexts, i.e. the data which may be exposed or protected can vary from application to application as well as from user to user even for the same application. In addition, service providers must be protected from malicious service users, i.e. it has to be ensured that the service user cannot get hold of the algorithms provided by the service provider.

In order to provide mutual protection between service user and service provider we introduce the concept of privacy guaranteeing execution containers. These are containers, which are independent of the service provider as well as independent of the service user. They ensure the following properties:

1. All data may be stored in volatile memory only and will be deleted after completion of service use; this has to hold true for service provider as well as for service user data.
2. The communication between the code executed in the container as well as the communication between the container and any third party is to be restricted to what is agreed between the service provider and the service user. This agreement is denoted as privacy contract throughout the rest of this paper.
3. The local exchange of messages and implicit communication e.g. via shared memory is prohibited.

If property (1) is fulfilled the container may be executed on any location (server or mobile) due to the fact that there is no way to get low level access to the data of the other side. The benefit is that load balancing becomes feasible. Computational expensive services do not have to be executed on the mobile device.

If property (2) is fulfilled there is no chance to steal data during the service use. The problem here is to define a set of allowed messages. On one hand it has to be sufficiently large to allow service fulfillment. On the other hand it has to be as restrictive as possible in order to ensure that it cannot be misused to steal data, and to enable the container to verify the content of the allowed messages.

If property (3) is fulfilled a service running in a container simultaneously with other services cannot share its knowledge about gained private data with other services that are concurrently executed within the PGEC. Hence it is not possible to extract private data via an additional service and a faked user with a very loose privacy policy.

The concrete behavior of the PGEC depends on the privacy preferences of the service user. The latter is negotiated between the service user and the service provider. The resulting document is called a privacy contract and defines which information may be accessed through the PGEC and which messages may be sent through the PGEC to which communication endpoints. In addition the restrictions that can be defined in the privacy contracts depend on the application/service which is run inside the container.

In order to make sure that the service user as well as the service provider will trust the PGEC, it has to be implemented by a trustworthy third party, and it has to be signed by that party using a PKI certificate e.g. from VeriSign.

3.1 Privacy Contracts and Their Dependency of Service Types

The communication between the code, which is executed inside the container and the service provider outside, is restricted by a privacy contract. The privacy contract has to be negotiated between the service provider and the service user. Appropriate means to do so have been proposed in [17]. Both parties have to sign that contract. It defines which kind of messages may be sent by the code providing the service and to which communication endpoints they may be sent. The rules that have to be defined inside the privacy contract depend on the kind of service. Up to now we identified three classes of services.

1) Logically delivering services
2) Logically controlling services
3) Physically delivering services

Logically delivering services do not need to communicate with the service provider side. A navigation service is a representative of this class. In order to provide its functionality, it only needs to read local service data such as a map and the current position of the mobile device. No communication with the service provider is needed, except in case these services are charging a small amount of money. Especially in this class we have to address the presentation of the service results. Presentation is usually considered as a display with a GUI or similar, and this can be misused to retrieve private data from inside the container via an allowed means. A virtual graphical device can actually write the information to disk as a straight forward attack. Services are most likely not presenting own secret data to the service users. The party that does not have actual control over what is displayed is the user. Hence, we need to differentiate between a container running at service provider's side suspected to fraudulently obtain private data and a container on the user's side who requires the private data to be protected. Thus we may allow services running in the container on the user's device to display any result or information and to prevent any audio or video generation on the service providers machine. The output device might be stated in the contract as well and could be authenticated by knowledge of the private key that was used to generate the contract signature.

The logically controlling services do not provide any benefit without a chance to send messages back to the service provider. Representatives of these logically controlling services are remote control services for cameras in scenic environments. These for example essentially need to send messages back home, such as move right, left, up and down. The privacy contract has to define a vocabulary for this class of services, which on one hand is rich enough to provide a comfortable handling and is on the other hand sparse enough to make sure that malicious code cannot use this vocabulary to encode privacy relevant information and send it back using the allowed messages. In addition the vocabulary has to be defined in a way that enables the communication interface of the container to check whether or not a certain message is allowed. We propose that the vocabulary consists of kinds of literals, i.e. the code can only send predefined messages. In this case the communication interface can use a simple string compare operation to verify whether the messages are allowed or not.

The last class we identified is the class of physically delivering services. Online-shops or print services are members of that class. Those require actually disseminating private data such as shipping addresses respectively content of pages to print. They require weakening of the privacy guarantee. Consider a shopping service that needs at least the shipping address given to the delivery service as well as information for clearing to be given to the bank. In case such information disclosure is agreed to in the privacy contract, the service requests the container to send the information to the appropriate party. Thus it is made sure that

only pristine information is given and to those parties only that the information was supposed for. This approach involves multiple service transactions, which are not described in detail in this paper. But note, from the moment this privacy relevant information leaves the container there is no control over it anymore.

3.2 Components of the Container

Figure 1 shows the components of the privacy guaranteeing execution container, i.e. the execution environment, the communication interface, the privacy contract and covert channel attack protection means. The first three are discussed in the following subsections, whereas the latter is omitted due to space limitations and since it is not needed to realize the core functionality of the container.

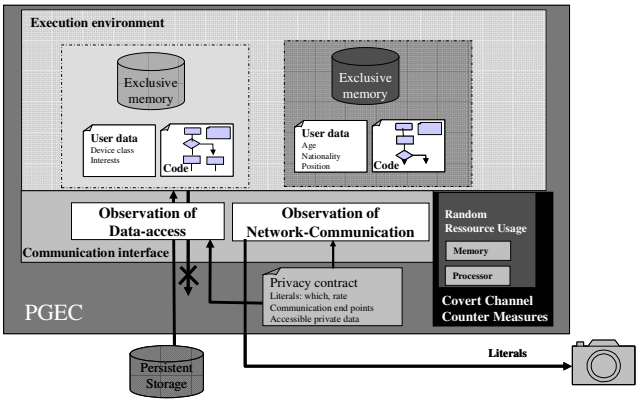

Figure 1: Components of the PGEC

3.2.1 Communication Interface

The task of the communication interface is to restrict access to data as well as the message exchange to what is agreed between the service provider and the service user.

3.2.1.1 Restricting Network Communications

The privacy contract states which messages may be sent to which communication endpoints, which data is to be retrieved from the user through the container and contains other privacy relevant information. The decisions taken by the communication interface are based on this privacy contract. In principle the communication interface is a kind of rule engine for which the rules are defined in the privacy contracts. In case the executed service in the container definitely needs to communicate with the outside of the container at its origin or any other party, this communication must be limited in a way that it prevents transmission of privacy relevant data. The communication must not be established by the service itself but merely by the container. An API to send messages is provided by the communication interface. The service has now the capability of initiating the sending of predefined messages by communicating via the communication interface of the container only. That way it can be ensured that the service does not send privacy relevant data to somewhere else. The destinations of the messages as well as the possible messages themselves are defined during the negotiation process of the privacy contract. The container checks the messages given through the API up on compliance with the privacy contract and sends the actual message to the specified communication endpoint or dismisses it.

Even if the vocabulary of the predefined messages is well defined, i.e. if it prohibits easy sending of sensitive data, it can be used for this purpose if an unrestricted number of messages may be sent. Hence, the container may also limit the frequency of such messages or it could deliver orthogonal messages in unspecified order. This decreases the chance of encoding information other than the control information to be transmitted by the message. Exemplary for such service, remote control services for electronic devices shall be named. E.g. an air conditioner does not need to know the privacy relevant information of someone's preferred room temperature, but only needs to be adjusted warmer or cooler until the measured temperature fits the personal preferences.

A service may need to receive additional information from its origin at the outside of the container. In this case the communication interface provides an API to request such information by response of a function call or by providing of predefined data streams. The container has the power to suppress messages from the services origin to the executed service. This prevents from guessing information and acknowledging by the predefined messages.

If a service tries to send data not agreed to in the privacy contract or to communication endpoints not agreed to a privacy exception is generated. Such privacy exceptions could also be logged. This log may help to prove fraudulent behavior during the negotiation process or to prove claims of reimbursement of unjustified service charges, if the service is not functional due to the lack of particular information.

3.2.1.2 Restricting Data Access

The second task is to control data access. Here again the privacy contract describes what is allowed and anything else is to be prohibited. In addition write operations to persistent storage have to be blocked by the communication interface.

Since the data needed is application/service dependent there has to be a specified way in which the container gains access to the data potentially passed to the services inside it. We identified two approaches.

The first approach is to specify an API to push information from the outside into the container. This enforces every application using the container to execute services to adapt to that API. The bigger problem comes with changing privacy relevant data that may be needed by services but not as often as they change. Thus, a data push approach would result in a never-ending push thread that takes up computational power, probably without any positive effect on the services in the container. If the data is pushed into the container the container itself has hold of these data and the garbage collector cannot take care of the demanded data deletion upon service completion.

The second approach is to give the container accessibility to the data at install-time or even at run-time. To accomplish that, we propose a data access component in the container as displayed in Figure 1, which grants read access only, and which uses the privacy contract to check which data may be read. The latter is not an essentially critical task since no data can leave the execution environment, if it is not allowed in the privacy contract, i.e. the

communication interface will filter all messages and other communication means are restricted as well.

3.2.2 Execution Environment

The execution environment is merely a logical construct that has to ensure that programs within can access the container interfaces but have no access to data or code in other execution environments. It provides the necessary infrastructure to services executed inside the container, e.g. access to the processor, volatile memory etc. It is also responsible for the cleanup operations that have to be done when a service is no longer used, i.e. it has to make sure that all data is really deleted. If different services running within the same execution environment on the same machine, it is the task of the execution environment to ensure that these services do not have any communication with each other. This means, even the use of a shared memory segment has to be avoided by the execution environment.

4. IMPLEMENTATION ISSUES

PGECs have to fulfill mainly three tasks. They have to provide an execution environment, to ensure proper clean up of data and code when the service is no longer used and to restrict the information exchange via all possible channels to what is allowed in the privacy contract. This functionality can be inherited by using runtime environments that ensure secure code execution such as Java runtime environment, .NET or even the Macromedia Shockwave Interpreter. All of the mentioned approaches provide some kind of a sandbox model which limits the access of foreign code to local resources such as file system and network. Since our PGEC needs to protect itself against the runtime environment additional conditions need to be fulfilled to guarantee correct behavior the PGEC.

4.1 PGEC and its Host Run Time Environment

The code is executed in a runtime environment and thus may only access system resources through that runtime environment. This circumstance allows limiting the access to resources by security managers. These security managers obey certain security policies. The code inside the sandbox may only access those resources that are explicitly granted. The limitation of the service permissions should go that far that they are only able to communicate with their execution environment. Thus the service is not able to send any information anywhere else but to the container. There is no need for a service to directly access any local resource. Everything the service is authorized to by the privacy contract can be accessed through the container. While the service providers may start a container component themselves they are actually enabled to set the security policies on their own. In order to still guarantee proper functionality of the PGEC the security policy that applies to the services in the container's execution environment must not be adjustable by the executor of the container. Thus, the according security policy must be fixed and an appropriate security manager must be running to obey this policy. We propose to let the container be this security manager including the constricted policy for the services inside. Our container is constructed such, that it does not instantiate if

1. it is not the first component at all started in the JVM,
2. there is already another security manager installed or
3. the JVM has been tampered with.

Condition (1) is used to ensure that no components are started prior to the PGEC. Otherwise it was possible for those components to open and keep a network or file handle without control of the container. Such might be used to circumvent the container access restrictions.

Condition (2) is needed to ensure that only the trusted and certified security manager of the PGEC is running. If there was another security manager installed beforehand it may not be possible to install the containers own security manager and the other manager cannot be trusted by the container, which results in a security and privacy leak.

Condition (3) ensures that the security features provided by the JVM are not circumvented by a malicious JVM implemented by an attacker. Such a JVM probably does not implement any of the security concepts built-in into Java. By that private data may be released even if the security manager of the PGEC is enabled. Hence the container has to check, whether the JVM it is running in is a well-known one. We are going to tackle this by hash checking of the components of the currently running JVM. This can ensure that the privacy container respectively its native components are running besides an approved JVM. Those native components may further hash check their callers from within the JVM to prevent Man-in-the-Middle approaches using an approved JVM. Code attestation approaches for similar purpose are also proposed by [14] and [15]. The secure hash checking also ensures that those classes accessing external resources are the ones that actually do the call for access checking at the security manager.

These three conditions ensure that no private data can be disseminated without user consent. They ensure that the PGEC runs only in a proper and secure environment and thus data is secured by PGEC means. Otherwise they prohibit container instantiation and by that data access since services may access privacy relevant data only inside the container.

4.2 Isolating Execution Spaces Inside the PGEC

Security permission checking is only done for external resources like files, sockets and native library linking but not for object access in memory. Thus, data exchange between services executed in different execution spaces by some shared memory - provided by static fields or methods in the services - must be prevented by means of the PGEC itself. Hence, we have to ensure that classes from one execution space may neither access objects nor classes of another execution space. This is accomplished by using different class loaders for different execution spaces, which ensures that objects in different execution spaces do not have handles on instances from other spaces. Only the container or classes within just that other execution space know those handles. The container will not give those handles to the objects instantiated in another execution space. To access static fields or methods of a class the accessing object must have hold of that class with static fields. If it is loaded by a particular class loader instance it will not get hold of the classes loaded by a different class loader instance but load an own copy of a class that name.

Thus, no static changes in instances of that class are visible to objects instantiated by the other class loader. Thus there is no shared memory available.

5. CONCLUSIONS

In this paper we have introduced the concept of privacy guaranteeing execution containers. The major benefit of this concept is that private data is not exposed to service providers. Thus, the user keeps full control over her data. This is achieved by providing an execution environment, i.e. the PGEC that is independent of the service user as well as independent of the service provider. The PGEC may access user data and also run code from the service provider. The PGEC has to be certified so that it is proven from a third party that the user data as well as the application code is deleted whenever the service is quit. Thus, for user data as well as for service provider code a kind of "one time use" is guaranteed. In addition the PGEC restricts communication to what is explicitly allowed by the service user. So there is no chance for malicious code to first copy user data and then sent it back home.

In order to illustrate the level of security that the PGEC concept provides we have discussed how modified execution environments can be detected. A properly implemented container can render those attacks useless, or at least it forces potential attackers to spend considerable effort to gain access to private data. So PGECs might become an economically one hundred percent secure solution.

The following key components of the PGEC are already implemented:

1. class loader for isolated execution spaces
2. security manager ensuring appropriate access from the container but not directly from the execution environments
3. hash checking of the runtime environment to detect tampering with the run time environment.

Our next research steps are the combination of the PGEC concept with our privacy negotiation approach as well as the extension of the latter. Up to now it does not reflect the need to define allowed messages and communication endpoints. A prototypically implementation of the complete PGEC is also planned for the next months. In addition we will investigate covert channel attacks and appropriate counter measures in order to further improve the security of our PGEC.

6. REFERENCES

[1] W3C: Platform for Privacy Preferences (P3P) Project, see http://www.w3.org/P3P/

[2] W3C: A P3P Preference Exchange Language 1.0 (APPEL1.0), W3C Working Draft 15 April 2002, http://www.w3.org/TR/P3P-preferences/

[3] J. Cuellar et. Al.: Geopriv Requirements, RFC3693, available at http://www.faqs.org/ftp/rfc/pdf/rfc3693.txt.pdf, last viewed May 2005

[4] P. Langendörfer, R. Kraemer: Towards User Defined Privacy in location-aware Platforms. Proceeding of the 3rd international Conference on Internet computing, USA. CSREA Press, 2002.

[5] M. Bennicke,P. Langendörfer: Towards Automatic Negotiation of Privacy Contracts for Internet Services. Proceeding of the 11th IEEE Conference on Networks (ICON 2003),. IEEE Society Press, 2003.

[6] W. Wagealla, S. Terzis, C. English: Trust-based Model for Privacy Control in Context-aware Systems, 2nd Workshop on Security in Ubiquitous Computing, Ubicomp, 2003

[7] K. Synnes, J. Nord, P. Parnes: Location Privacy in the Alipes platform. In Proceedings of the Hawai'i International Conference on System Sciences (HICSS-36), Big Island, Hawai´i, USA, January 2003.

[8] M. Gruteser, D. Grunwald: Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking, ACM/USENIX International Conference on Mobile Systems, Applications, and Services (MobiSys), 2003

[9] J. Lopez et al: Access control Infrastructure for Digital Objects, International Conference on Information and Communications Security (ICICS'02), pp.399-410, LNCS 2513, Springer-Verlag, Singapore, December 2002.

[10] Cryptolope in Ricardo Haragutchi: IBM Redbook Building the Infrastructure for the Internet http://www.redbooks.ibm.com/redbooks/pdfs/sg244824.pdf

[11] Antonio Maña, Javier Lopez, Juan J. Ortega, Ernesto Pimentel, Jose M. Troya A Framework for Secure Execution of Software, International Journal of Information Security, Volume 2, Issue 4, pp.99-112, Springer, November 2004.

[12] H. Garcia-Molina, S. Ketchpel; N. Shivakumar: Safeguarding and Charging for Content on the Internet, Proceedings of Int. Conf. On Data Engineering, 1998

[13] S. Yamada, E. Kamioka: Access Control for Security and Privacy in Ubiquitous Computing Environments, IEICE TRANS. COMMUN., VOL.E88-B, No.3 March 2005

[14] A. Seshandri, A. Perrig, L. van Doorn: Using Software-based Attestation for Verifying Embedded Systems in Cars, Embedded Security in Cars Workshop (escar), November 2004

[15] http://www.trustedcomputinggroup.org

[16] Handbook for the Computer Security Certification of Trusted Systems, Chapter 8, NRL Technical Memorandum 5540:062A, 12 Feb 1996

[17] M. Maaser, P. Langendörfer: Automated Negotiation of Privacy Contracts, Proceedings of the 29th Annual International Computer Software and Applications Conference (COMPSAC), IEEE Society Press, 2005

[18] Huda, N., Yamada, S., Kamioka, E., Privacy Protection in Mobile Agent based Service Domain, Proceedings of the Third International Conference on Information Technology and Applications (ICITA'05), Sydney, July 4th- 7th 2005

[19] A. Yannopoulos, Y. Stavroulas, N. Papadakis, D. Halkos, T. Varvarigou, A method which enables the assessment of private data by an untrusted party using arbitrary algorithms but prevents disclosure of their content, In P. Langendoerfer, V. Tsaoussidis (eds.), Proceedings of the 3rd International Conference on Internet Computing, CSREA Press, 2002