Distributed Service Environment (Smart Spaces) Security Model Development

Kirill Yudenok, Kirill Krinkin
Saint-Petersburg Electrotechnical University
Russia, St.-Petersburg
{kirill.yudenok, kirill.krinkin}@gmail.com

Abstract

Access control mechanisms play a key role in many areas of computer science, however, for the information provided on the basis of semantic web and established solutions don't exist. This work focuses on the research in this area, in particular to ensure the information security in distributed service environments (smart spaces), which are the most promising application of standards and technologies of semantic web.

The main focus of this paper will be devoted to the analysis and investigation solutions to develop security model and mechanisms for a smart space platform, as well as its comprehensive testing. As a test platform was chosen Smart-M3 platform, which has the highest degree of elaboration and maximum prospects for further applications.

Index Terms: Smart spaces, Smart-M3, HIP, Security, Access control.

I. INTRODUCTION

Modern trends in information and communication technology led to the need for a stable and reliable infrastructure for storing and retrieving various kinds of information from a wide range of actors in the information environment. This infrastructure is called "*smart space*" (SS). Systems that implement the function of smart space, suggest the presence of multiple devices using a common representation of the available resources and services. Through the use of smart space can provide better support to the user that provides the flexible usage and inclusion in the intellectual environment of the various new devices as well as access to information and services from any device such environment, regardless of their physical location. The main problem of intellectual environment consistency devices is that the resources of the environment are shared among different devices and retrieve information not necessarily performed from the device on which it is stored [1].

More modern application platform of smart spaces take a common approach of shared memory and increase interoperability through the use of ontologies, graphs based on the data model of the semantic web. But in order to develop freely and maintain a controlled situation, highly dynamic applications, shared memory infrastructure of smart space must include access control mechanisms, at least, to provide exclusive access to the selected part of the information that must be updated.

Access control of digital systems has been deeply studied since the emergence of computer science at all levels of computer architectures abstraction. Definition and provision of complete and common access control model that is optimized for data based on graphs is a complex task that is beyond the scope of this paper [3].

The focus of this paper is focused on the research and analysis of the subject area, a description of possible solutions for the creation of the security mechanisms and the development the security model and mechanisms for the smart spaces Smart-M3 platform.

II. SMART SPACE SMART-M3 PLATFORM

A. Overview of smart space Smart-M3 platform

Smart-M3 is an open source software platform [2] that aims to provide Semantic Web information sharing infrastructure between software entities and various types of devices. The platform combines ideas of distributed, networked systems and Semantic Web [4]. The major application area for Smart-M3 is the development of smart spaces solutions, where a number of devices can use a shared view of resources and services [5]. Smart spaces can provide better user experience by allowing users to easily bring-in and take-out various electronic devices and seamlessly access all user information in the multi-device system from any of the devices.

The simplified version of the Smart-M3 smart spaces reference model is shown in Fig. 1. The Knowledge Processors (KPs) represent different applications that use the smart space. At figure KPs is shown as M3-agents. The smart space core is implemented by one or several Semantic Information Brokers (SIBs) interconnected into the common space. The information exchange is organized through transfer of information units (represented by RDF triples, resources in the form of subject-predicate-object expressions) from KPs to the smart space and back. The information submitted to the smart space becomes available to all KPs participating in the smart space. The KPs can also transfer references to the appropriate files/services into the smart space, since not all information can be presented by RDF triples (e.g., a photo or a PowerPoint presentation). As a result the information is not really transferred but shared between KPs by using smart space as a common ground [4].

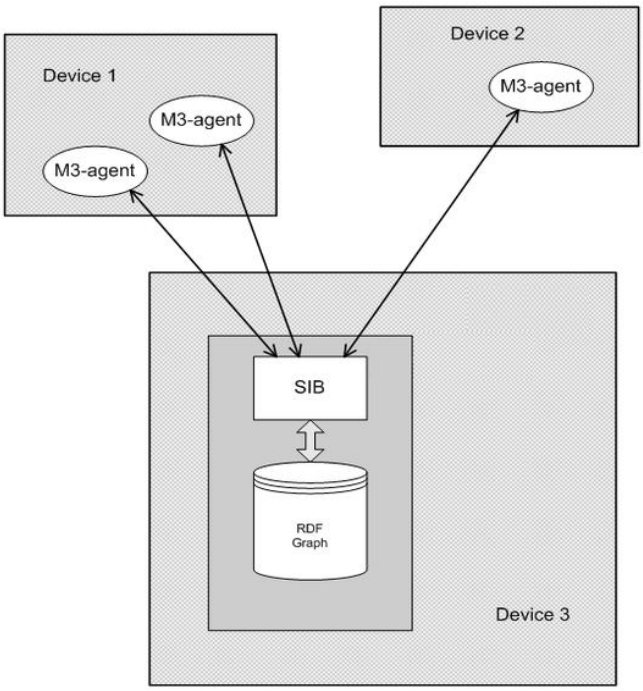

Fig. 1. Smart space based on Smart-M3: simplified reference model

B. Security problems of Smart-M3 platform

The current version of the Smart-M3 platform has some access control mechanisms for information of the smart space, but at a lower level of abstraction [8]. Investigation of these mechanisms is conducted in order to find an optimal and fastest way to control access in restricted conditions (width of channel, the performance of the computing device, etc.). The usage of proven technologies will avoid some of the difficulties and realize the long-unsolved problem.

The main security problems of Smart-M3 platform are:

- Absence of identification and authentication mechanism of smart space client. Clients connection without the identity confirmation to the smart space. Any agent may always connect to the space, knowing only its connection settings.
- Absence of smart space users access control and authorization mechanism. At this moment, every user has the same rights in all smart space, all operations, such as insert, update, delete, query and subscribe are available on initial connection.
- Absence of privacy, because all the information is stored in the public domain.

Based on the problems described above, we can conclude that the platform does not have the basic security mechanisms, which are a major threat not only for the data stored in the space, but also throughout as a whole platform.

III. SMART SPACE SECURITY MODEL, MECHANISMS AND ACCESS CONTROL DEVELOPMENT

A. Smart space security model and mechanisms description

Domain and its characteristics analysis showed that the concept of smart spaces is a new paradigm in software development, but security issues are relevant in this case, which has led to the creation of standard security mechanisms in the new environment.

In the field of computer security there are basic models and methods for ensuring the security of computer systems. Each system must meet the set of security requirements for it works in any conditions of its life cycle.

Select the main smart space security mechanisms that necessary to be developed. To access control to the smart space it is necessary to develop mechanisms for identifying and authenticating SS clients that allow monitoring and preventing unauthorized access to the SS.

To protect the integrity and data access should provide an authorization mechanism for the smart space subjects to control access to SS data based on access rights assigned to each subject.

Creation of security mechanisms lies entirely on the shoulders of the system designer, but there are waste solutions that to be followed in the design of security computer system.

At this stage various methods of security mechanisms in the SS area were analyzed and investigated from standard security models to third-party solutions offered by developers of security systems.

B. Identification and authentication of smart space subjects

For identification and authentication of SS clients, it was decided to use the HIP protocol – Host Identification Protocol. This protocol presents some security methods, such as authentication, encryption and privacy.

1) HIP: HIP (*Host Identity Protocol*) is a host identification technology for use in IP networks such as the Internet, developed at the IETF since 1999 and reaching their first stable version in 2007.

The Internet has grown tremendously over the past twenty years and become a part of life for millions of people. The basic TCP/IP technology has served us very well. However, important issues such as mobility of Internet hosts over separate IP networks and simultaneous connections to several networks were not a part of the original Internet design. Furthermore, when the Internet grew from a small university network up to a global communication infrastructure, many security issues became apparent. The lack of reliable host authentication has prevented deployment of existing IP mobility extensions. Often, public Internet servers face Denial-of-Service (DoS) attacks that make the service unavailable to other users. [10]

HIP is designed to solve the following issues:

- mobility of Internet hosts over separate IP networks and simultaneous connections to several networks;
- reliable host identification and authentication;
- host mobility and multihoming;
- security and privacy over IPv4 and IPv6 networks.

HIP is developed to address these issues in an integrated approach that fits well within the TCP/IP architecture.

HIP enhances the original Internet architecture by adding a name space used between the IP layer and the transport protocols. This new name space consists of cryptographic identifiers, thereby implementing the so-called identifier/locator split. In the new architecture, the new identifiers are used in naming application level end-points (sockets), replacing the prior identification role of IP addresses in applications, sockets, TCP connections, and UDP-based send and receive system calls. IPv4 and IPv6 addresses are still used, but only as names for topological locations in the network. HIP can be deployed such that no changes are needed in applications or routers. Almost all pre-compiled legacy applications continue to work, without modifications, for communicating with both HIP-enabled and non-HIP-enabled peer hosts. [8]

HIP assigns to host a unique address using the private key (HIT). This address is permanent and does not change the host position in the space, when the network address is the same. If the network uses the HIP protocol, all IP addresses are replaced by cryptographic identifiers.

HIP protocol has built-in security mechanisms and connection privacy. Using the private key can be authenticated the host on the server. The server must also be configured to support HIP. For authentication, the server checks host HIT with private key.

HIP authenticates and protects connection between two hosts. HIP authenticates host and verifies the symmetric key between them for secure data transfer. The flow of data between end hosts is encrypted by IPsec ESP with a symmetric key, sets by HIP protocol. [9]

HIP architecture is shown at Fig. 2.

As a HIP implementations can be considered the following solutions:

- The *OpenHIP* implementation has been started by the Boeing Phantom Works company in the USA. It is run on Linux, Windows and Mac OS. The implementation is released with a GPL license. (http://www.openhip.org).

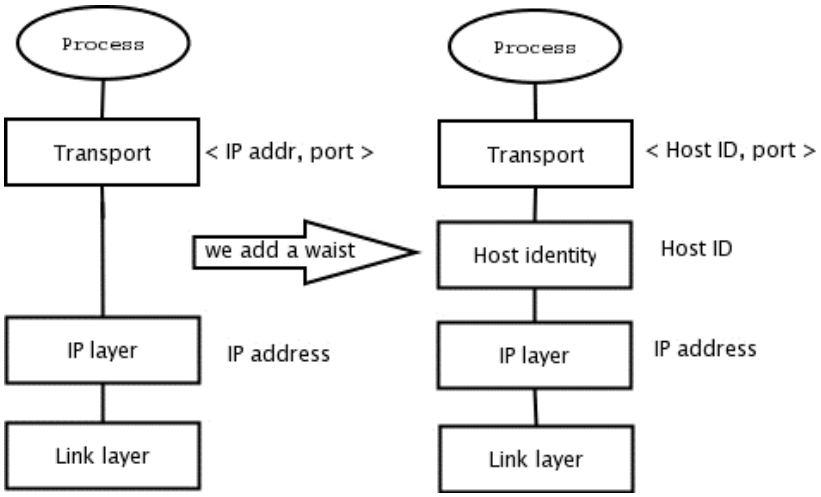

Fig. 2. HIP architecture

- The *HIP on Linux* (HIPL) implementation is developed by the Helsinki Institute for Information Technology (HIIT) in Finland. HIPL runs on Linux, and there is an ongoing project to port it to Symbian OS. For public use, the implementation is released with the GPL license, but a less restrictive MIT license is available on special request. (http://infrahip.hiit.fi).
- The *HIP for inter.net* implementation is developed by Ericsson NomadicLab, located in Jorvas, Finland. The primary platform for HIP for inter.net is FreeBSD, although the Linux variant is also available. Both IPv4 and IPv6 protocols are supported. The implementation has a special "Ericsson Finland Public Source" license. (http://www.hip4inter.net/).
- Also, there is an open-source but outdated HIP implementation in Python language by Andrew McGregor [10].

C. Authorization and access control of smart space subjects

Authorization mechanism and access control of smart space subjects can be constructed using the following solutions: discretionary security model, SS RDF-graph mapping as a virtual file system (VFS), named graphs, access control ontology [18, 19, 20] and some security extension for the SS database. SS RDF-graph mapping as a virtual file system used in this work, as the main mechanism for authorization and access control.

Mechanism of RDF-graph mapping to a virtual file system is very similar to the discretionary security model because of the binding to the file system. Mapping model as well as the discretionary model will have the access control list (access matrix) to distinguish rights between the system subjects. This model can be easily extended to the role. Can be concluded that the mapping model like a copy of the discretionary security model with the addition of different data represent algorithms as entities of the file system. Mapping RDF-data to the virtual file system bundle with its access control mechanism pushes us a little from the principles of distributed systems, but it fits into the framework of developing secure systems using established mechanisms [11].

A further step in the development of the RDF-graphs is named graphs, which have their own security mechanisms. Ontology access control can also be used, but due to continued access, performance of the entire system will be open to question.

Consider the mechanisms of a SS RDF-graph mapping as a virtual file system and named graphs in more detail.

1) SS RDF-graph mapping as a virtual file system: Providing a SS access control and authorization mechanism can be achieved by SS RDF-graph mapping as a virtual file system, which has its own established mechanisms of access control. All information of smart space is presented in a triple form in RDF-similarity graph, which is stored in the SS database.

Consider that the smart space (a set of "SIB-DB") is a file system (directory structure), which has a certain access rights "*RWX*":

- *R* - read a triple, extract its components *(S, P, O).*
- *W* - write (insert) the relation to the triple.
- *X* - a list of entities relationship.

Considering a smart space as a virtual file system, we assume that the RDF-graph will be as a directories tree and conclude that the directory access operations are analogous to RDF operations. For example, the "execution" right of file system in the RDF field can be represented as "get a list of all objects (relations) of this directory" or "list the entity relationships." Thus, projecting the operation (rights) of the file system to the RDF operations, we have a new area to solve the problem of access control to the information provided in the SS.

The implementation of RDF-graph mapping as a virtual file system can be realized with the FUSE technology, which allows developing your own virtual file system [12, 13].

2) Named graphs: A named graph is an RDF graph which is assigned a name in the form of a URI. A named graph is an entity with two functions *name* and *rdfgraph* defined on it which determines respectively its name, which is a URI and the RDF graph that it encodes or represents. These functions assign a unique name and RDF graph to each named graph. In this way, a named graph is a resource, identified by its name and so it can be described in the usual open way using RDF. Named Graphs are backward compatible with RDF. Named graphs can be used with any of the various levels of semantic theories for RDF: simple, RDF, RDFS or data typed interpretations from, or OWL Full interpretations from. There are languages to describe the named graphs (TRIX, RDF / XML and TRIG) and query languages (RDFQ, TriQL, SparQL) to work with them.

Named graph extends RDF-graph by adding new entities in the graph, such as "an authority", "a relationship of authorizing" and "warrant". An authority is a 'legal person'; that is, any legal or social entities which can perform acts and undertake obligations. The *'authorizing'* relationship holds between an authority or authorities and a Named Graph, and means that the authority in some sense commits itself to the content expressed in the graph. A warrant is a resource which records a particular propositional stance or intention of an authority towards a graph.

These entities are responsible for graphs availability, digital subscription, authorization attitude and trust policy. It also allows you to track all transactions performed with the graph. Construction of a SS named graph by adding new entities (warrant, authority) to the SS graph will use the techniques of trust and graphs availability and their information [15, 16, 17].

IV. THE IMPLEMENTATION OF A SECURITY MODEL AND ACCESS CONTROL PROTOTYPE BASED ON SS PLATFORM

This section discusses selected technologies, describes solutions and implementation of the security mechanisms in a smart space, such as identification, authentication and authorization.

A. The mechanism of identification and authentication based on HIP protocol

To add HIP support to the smart space Smart-M3 platform the following solution was produced – to develop a special agent on the server side of Smart-M3 platform, whose task is to identify and authenticate the SS clients. As well as the correct HIT-IP mapping binding setup in the system for work a HIP protocol with Smart-M3 platform.

1) HIP-agent development: When a client (KP) connects to the smart space (SIB), the connection is intercepted by HIP-agent, which identifies and authenticates the client based on the HIP protocol. Client and server sides must be configured to work via HIP.

Client communicates with HIP-agent using the Standard Socket API. When connected, the client sends the hash key on the basis of which is made a decision on the identification and authentication of the smart space client. If the hash is valid, the agent connects the client to the space (SIB). This solution is shown in Fig. 3.

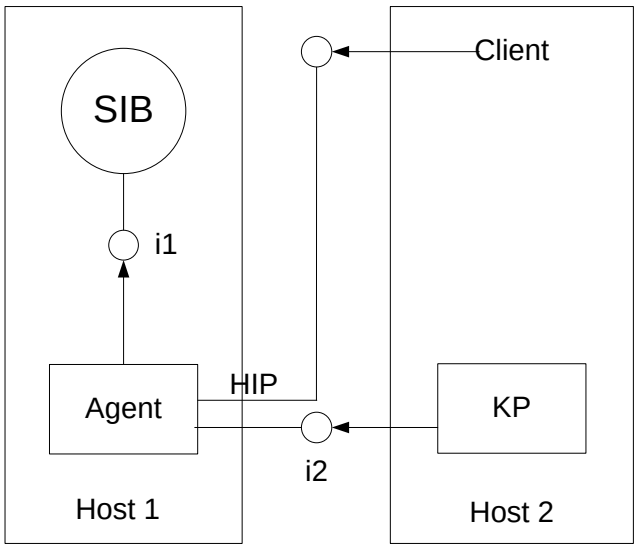

Fig. 3. HIP-agent for Smart-M3 platform

The process of the client connection to the space is shown in sequence diagram Fig. 4.

2) Setting the binding of HIT-IP mappings: To address this issue, there are several solutions:

1. setup bindings locally at the host, e.g. *hosts* files or *hipd* configuration file;
2. implementation of own application-specific resolution system;
3. using the tools provided by HIPL;
4. using of experimental HIP libraries (libhipl extensions).

As the solution was chosen to use the configuration tools provided with HIPL, namely tools like *hipconf* and *hipdnsproxy*.

hipconf is a user-level program that is designed to create new IDs and adding them to the main database of host ID (HI), setting the necessary parameters, routes and obtain the necessary information.

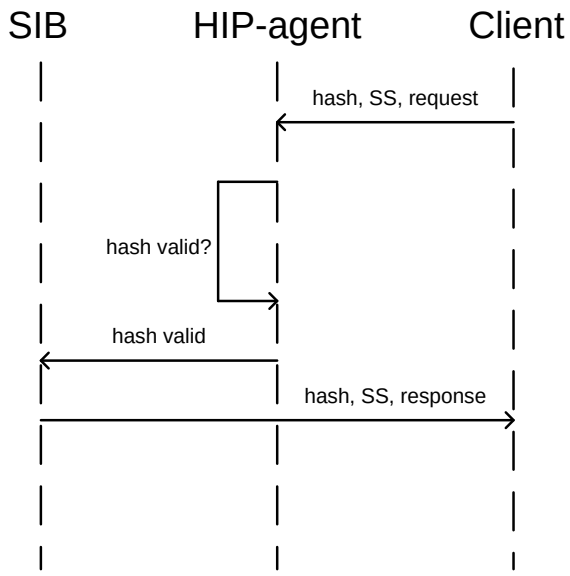

Fig. 4. The process of client connection to the space

hipdnsproxy lets you configure DNS on the client machine. There are two scenarios to set up bindings:

- forward DNS resolution (hostname → HIT+IP) using the DNS proxy that works with legacy applications;
- reverse DNS resolution (HIT → IP).

The first option works with "*vanilla bind*", that is, you can set up your own HI records in DNS. The second option requires a bind patch and it is not really needed in all scenarios. Configuration details HIP-IP bindings can be found in the official documentation (chapter five) [9].

B. SS clients access control and authorization mechanism

1) SS RDF-graph mapping as a virtual file system: Currently, Smart-M3 platform works as follows. On the server side is hosted semantic information broker (SIB) and the smart space database. To work with the client side, exist a special SSAP protocol, which is designed to exchange data between the parties.

All information of the smart space is stored in relational data storage, smart space database. Smart space information is presented in the triple form. The totality of these triples stored in a specially defined database tables of the Smart-M3 platform. All tables of the smart space database can be found in SQL-script "*createDB.sql*" of piglet_m3 module of Smart-M3 platform.

As the database serves embedded lightweight relational database - SQLite. SQLite stores the entire database (including definitions, tables, indexes, and data) in a single standard file on the computer on which the platform is performed. SQLite library is written in C and is a good API for working with the database [21].

After some research described in paragraph "*Authorization and access control of smart space subjects*" it was decided to develop a virtual file system that mapping smart space information that is stored in a database table to a certain directory structure for subsequent operations on them.

The directory structure of the mapping file system is shown in Fig. 5.

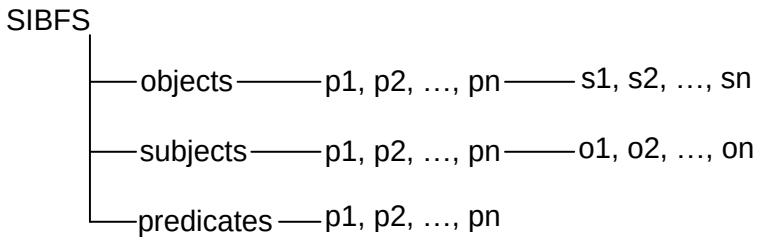

Fig. 5. File system directory structure

The directory structure has a structure that allows you to receive information in the form of "subject (s), predicate (p), object (o)", vice versa and all predicates (relations).

For a more accurate setting of access rights to the smart space triples (information) was designed updated directory structure, which is presented in Fig. 6.

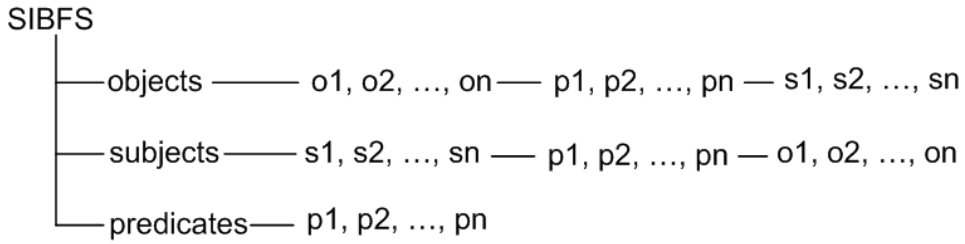

Fig. 6. Updated file system directory structure

Any Linux file system provides standard attributes to control access to files, directories and other entities of file system. Thus a file system provides the standard access control mechanisms for the users of the system.

The place of graph mapping as part of the platform is shown in Fig. 7.

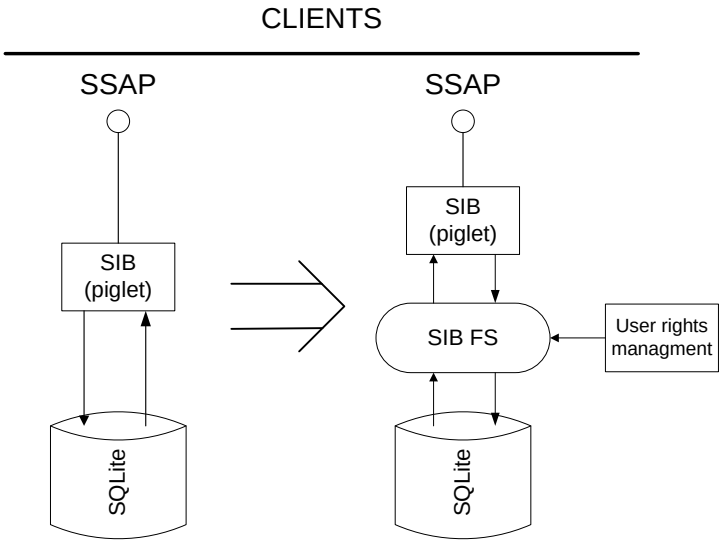

Fig. 7. The place of graph mapping as part of the platform

2) A smart space virtual file system creation: To develop a virtual file system was used FUSE technology (fusekit). [14] FUSE allows you to develop a fully functional file system, which has a simple library API, which can be used by non-privileged users and provides a secure implementation.

Smart space RDF-graph can be mapped in a VFS from smart space RDF-data and by SSAP protocol, because the exchange of data between the client and smart space is using it. In this case, it was decided to make smart space RDF-graph mapping from smart space RDF-data. After platform starting, entire smart space database will be stored, for example, in the file "*X*" of the user's home directory.

After the installation and configuration FUSE, you can create your own virtual file system.

General algorithm for the graph mapping process can be represented as follows:

1. Logic implementing to work with the smart space database:
   - obtaining all objects, subjects, predicates and their values from the smart space database;
   - record the data in the memory or data structure.

   This step is to create methods for working with the smart space data. Table "*triple"* stores all smart space triples in the following form, *s* - the subject, *p* - predicate, *o* - the object and *src* - triple information: TABLE triple (s INTEGER, p INTEGER, o INTEGER, src INTEGER); Table "*node*", in the *str* column store their values in accordance with the field *id*, number of s, p, o from *triple* table. You need to extract all triples values from the *node* table over the keys of the *triple* table and store them in memory or data structure.
2. A virtual file system directory structure creation on the basis of the data.

   FUSE (fusekit) allows you to create your own directory structure with files and other FS entities, set the permissions and other characteristics. At this stage it is necessary to create a virtual file system directory structure that is shown in Fig. 5.

After the graph mapping will gets a virtual file system with all smart space information (triples), which also is the primary access rights managing mechanism of all smart space clients.

Setting permissions on the directory structure of the VFS is a manual process. This method has the disadvantage of the administration as well as the installation of the access rights for each table or directory tables require careful control of the process.

To correct this deficiency it is planned to develop a special access rights managing tool. The main objective of this tool to install the access modifiers (labels) on the file system elements of the mapped graph (files, directories). This principle allows pointing rights to each triplet, whole group of triplets or individual relationship of triplets.

3) Implementation of a mapping model to the Smart-M3 platform: The process of implementing the mapping model to the Smart-M3 platform is as follows:

1. *piglet* module modification of the Smart-M3 platform:
   - 2.1. piglet proxy creation for new extensions;
   - 2.2. replacement of all database operations to the mapped file system operations;
   - 2.3. identification and verification of access rights of clients;
2. testing operations on the client side.

After the implementation of the smart space mapping mechanism to the platform architecture, we can control the information access process for all of its clients.

Eventually it will completely abandon the smart space database and replace it with a file access (virtual file access).

V. TESTING AND ANALYSIS OF A MODEL

A. *Functional testing of a model*

Testing of a designed security mechanisms held in a specifically deployed environment. As main operating system used Ubuntu 10.04 with installed Smart-M3 platform, realization of HIP — HIPL and other components necessary for the development.

1) HIP testing: To complete HIP work in the system, make sure that is the correct mapping of IP addresses in the HIT. To do this, configure *hipd* configuration file at both sides or use the utilities *hipconf* and *hipdnsproxy*. All the main mapping options listed in the "*Configuring Bindings HIT-IP mappings*" chapter.

After setting up HIP protocol, the system will use a special cryptographic keys (HIT) instead of IP addresses or jointly. To test the HIP in the systems you may try to ping second HIP configured host or use the HIP-compatible software, such as Sendmail, VLC, OpenVPN and others, but they also require a preset configuration [9].

HIP protocol setting on the client and server can communicate over a secure channel with a preliminary identification and authentication of the connected host's over the HIT key. It is completely transparent to the transport and application layers.

2) The SS RDF-graph mapping testing: During testing mapped file system, make sure that the SS RDF-graph is correctly mapping in the virtual file system.

A common scenario of testing mapping graph:

- Smart space deployment, Smart-M3 platform launch, content of SS with data.
  After platform starting will create a database of the SS, the file "X" in the user's home directory. As the SS data can be a specific ontology or set of triples.
- SS RDF-graph mapping in VFS mechanism launch: *$./sibfs ~/sibfs*.
  The result of mechanism launch is that from SS RDF-data will form a virtual file system to a predetermined directory structure (Fig. 5) and VFS will be automatically mounted in the system, for example in the "*sibfs*" directory.

In future this VFS is used to set the access rights for a certain groups of triples (directories and files of VFS).

B. *Analysis of the mechanisms for compliance with the requirements. Detection of model flaws and a description of their possible solutions*

HIP protocol meets all the requirements for the creation of identification and an authentication mechanism based on a special agent of the platform, but has some difficulty in configuration, in provided API and is not supported by all operating systems and has bugs in some of its components (*hipdnsproxy*).

SS RDF-graph mapping mechanism to a virtual file system also fulfills its requirements, but not perfect, such as the difficulty in setting up and monitoring of access rights for all users of the system, but can be solved by creating a special configuration access rights utility.

It is planned to replace the authorization mechanism for implementation using named graphs that will solve some problems of graph mapping.

VI. CONCLUSION

The research and development of the security model and mechanisms in distributed service environments (smart spaces) achieved the following results:

- analyzed and designed the HIP protocol-based mechanism of identification and authentication;
- the mechanism of authorization and access control SS subjects by mapping RDF-graph on the file system is developed; mechanism tested in the Smart-M3 platform;
- the process of implementation mapping mechanism to the Smart-M3 platform is started.

As a result of the project was examined the basic security and access control models, researched and designed the main methods to develop their own security mechanisms in smart spaces, designed and tested mechanism for SS RDF-graph mapping to the virtual file system in the Smart-M3 platform.

VII. Future Work

The next step in the improvement of the security model and mechanism will be:

- implementation of mapping model to Smart-M3 platform;
- set permissions tool development for mapping FS;
- HIP-agent development;
- named graph authorization system development;
- adding developed mechanisms to new version of Smart-M3 platform (Redland).

A working implementation of the security prototype mechanisms for the Smart-M3 platform to be developed for the 13th FRUCT conference.

Acknowledgment

Authors would like to thank FRUCT Smart Space Working Group (SS WG) for providing feedback and guidance.

References

[1] N.G. Shilov, A.M. Kashevnik, "Modern system of interaction of mobile devices in smart spaces: requirements and technologies", UDK 004.8 (rus).
[2] Smart-M3 Open Source Project, http://sourceforge.net/projects/smart-m3
[3] A. D'Elia, J. Honkola, D. Manzaroli, T. Salmon Cinotti, "Access Control at Triple Level: Specification and Enforcement of a Simple RDF Model to Support Concurrent Applications in Smart Environments", RuSmart, August 2011.
[4] J. Honkola , H. Laine, R. Brown, O. Tyrkkö, "Smart-M3 Information Sharing Platform", *1st Workshop on Semantic Interoperability in Smart Spaces*, 2010.
[5] RDF Semantics, http://www.w3.org/TR/rdf-mt/
[6] J. Honkola, H. Laine, R. Brown, I. Oliver, "Cross-Domain Interoperability: A Case Study", *Computer Science LNCS*, vol. 5764, pp. 22-31, 2009.
[7] B.A. Galatenko, "Categorization of information and information system. Provide a basic level of information security", 2006. (http://citforum.ru/security/articles/categorizing/). (rus)
[8] P. Nikander, A. Gurtov, T. Henderson, "Host Identity Protocol (HIP): Connectivity, Mobility, Multi-homing, Security, and Privacy over IPv4 and IPv6 networks", *IEEE Communications Surveys and Tutorials*, 12 (2), 2010.
[9] Official HIPL manual: http://infrahip.hiit.fi/hipl/manual/index.html
[10] A. Gurtov, Host Identity Protocol (HIP): Toward the Secure Mobile Internet. *HIIT*, Finland, 2008.
[11] P.N. Devyanin, "Mathematical foundations of computer security. The application", *Institute of Cryptography, Telecommunications and Computer Science*, Moscow, Russia, November 2009. (rus)
[12] FUSE documentation: http://www.ibm.com/developerworks/ru/library/l-fuse/, http://fuse.sourceforge.net/
[13] FUSE sources: http://sourceforge.net/projects/fuse/files/fuse-2.X/
[14] Fusekit sources and documentation: http://code.google.com/p/fusekit/
[15] J.J.Carroll, C.Bizer, P.Hayes, P.Stickler, "Named graphs", 2005.
[16] J.J.Carroll, C.Bizer, P.Hayes, P.Stickler, "Named graphs, provenance and trust", 2005.
[17] T.Gibson, K.Schuchardt, E.Stephan, "Application of Named Graphs Towards Custom Provenance Views", 2009.

[18] P.A. Lomov, M. G. Shishaev, "Developing an ontology for semantic access control", *The establishment of the RAS Institute of Informatics and Mathematical Modeling Process Biological Institute*, Russia, 2010. (rus)
[19] B. Andersen, F. Neuhaus, "An ontological approach to information access control and provenance", *OIC*, 2009.
[20] A. Mohammad, G. Kanaan, T. Khdour, S. Bani-Ahmad, "Ontology-Based Access Control Model for Semantic Web Servives", *JICS*, p. 177-194. England, UK.
[21] SQLite documentation: http://www.sqlite.org/docs.html