# Secure Agent Architecture for Wireless Devices

Mgr. Michal Laclavik, Dipl. Ing. Zoltan Balogh, Dipl. Ing. Ladislav Hluchy CSc.
Slovak Academy of Sciences, Institute of Informatics
Dubravska cesta 9, Bratislava
Slovakia

## ABSTRACT

In this paper we are giving proposal for secure agent architecture for wireless devices. We focus mainly on security protection from the system itself as well as a different security view on distributed services and protection of data passed to those services. In addition, we describe possible applications of the architecture and implement fundamental parts of the architecture.

**KEY WORDS:** software agents, architecture, wireless, security

## 1. INTRODUCTION

Everyone can see how cell phones and other mobile and wireless devices are taking their place in an every day life of people. The wireless devices are becoming more and more sophisticated, multifunctional and advanced. We think that the next step will be creating software platform for cell phones. This way, different software makers can start developing programs for cell phones and users can better customize their devices and its usage. Such platforms have been already developed for Palms, Pocket PCs and also some cell phone communicators such as Nokia 9210. None of them is agent-based platform. We will try to explain recent security problems in non-agent platforms.

Security is on a very good level in today's computer or wireless platforms. However, security against software makers' impacts is not solved. Any software installed on our device can possibly take control over our data and misuse them. When passing data through the Internet to some Internet application, no one can steal our data over the network but we cannot be sure what application on other site will do with the data. In other words, security of communication is all right but we have to trust software makers on each side of communication.

Open Source or any software where code is available is a great solution for trusting software on your device side. You can check source code for any Trojan horses or security holes. Now what about other side of the communication pipe?

As we mentioned, agents are the best solution for solving other side software security problem. Our solution is simple. Software on other side will come to our device as an agent, device will lock it inside and service of the agent will be provided. Naturally, not all applications can be solved using this method but most of them can. We will discuss it more in the proposal. Proposed architecture can be used also for non-wireless systems such as PCs, Clusters or any Internet and intranet based systems. We chose wireless platform because, according to our vision, it is easier to bring new technology into wireless world. Wireless technology for mobile phones, for example, depends on a few cell phones producers (Nokia, Ericsson, Motorola, Siemens, etc.). Moreover, bringing all changing technology into PCs is not easy and almost impossible to use. However, we will work on device independent architecture in the future.

## 2. SECURITY IN RECENT MULTI AGENT SYSTEMS (MAS)

Security is very important issue in all the systems. Many people can see and think that security is not solved yet in any Internet Based System. Security holes and successful hackers impacts occur very often in the Internet world because of programming mistakes or human failure. Theoretically we can say "Security has been already solved" and that is true. The problem is that it has not been proposed and implemented yet in many systems. We can divide security to several levels [7]:

- Security of communication
- Security of system against outside impacts
- Access rights
- Approving users, agents and others
- Security against inside software and other side software

*Security of Communication.* We can provide this by using SSL what is a standard encryption method used for example for Internet Banking. Securing of communication in MAS is described in our Security proposal. KQML [4][5] is used as communication language in our experiments and our proposal.

*Security of System Against Outside Impacts.* Choosing a right and secure platform with installed security patches can solve most of those problems. In addition, some access restrictions must be set up. In MAS based on Java, implementing a good security manager can solve this. [2][5]

*Access Rights.* A very important thing is securing against inside impacts. We need to define permissions for a certain level for people, agents etc. Also its communication with system has to be encrypted by public private key method.

*Approving Users, Agents and Others.* Approving someone, who communicate is important. Even if we are securing communication using the public-private key method, we want to be sure that agent on the other side is the one, which we expect to be. Certification authorities take care of this.
In our secure communication proposal, [11] central or distributed database of agent public keys (DPK) is taking this place. Each Agent Place or each agent who has a public key has to have this key stored in this DPK with its information. When new agent gets created, the public key is generated and sent to DPK by its creator with its information. DPK Security Agent can represent DPK. Each agent has standard method to access DPK agent by secure connection to get confirmation about public keys of other agents.

*Security Against Inside Software and Other Side Software* is the only unsecured spot in today's systems. This article tries to answer this problem.

## 3. PROPOSAL FOR AGENT ARCHITECTURE FOR WIRELESS DEVICES.

*Overview of the problem*
There are various applications of Intelligent Agents. MAS already supports communication, security managers, or secure migrating. What is not supported is protecting agents and information against system itself.

Let us describe those problems:
Here is an example: somewhere on the Internet is a service (data + application)
We will access this application by Internet browser. Application will work on https protocol in a way all communication between user and application is secured.

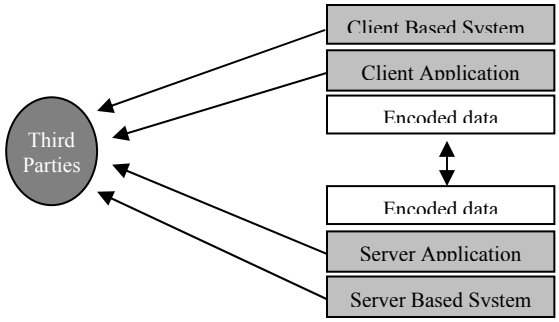


Figure 1

As you can see on the Figure 1, Client System or Applications as well as Server System and applications can send any data to the third parties over the network.

In our proposal we are trying to secure those unsecured spots.

*Proposal*
Agent technology is suitable for this because service provided by a service provider can be brought as an agent to our device and act there. Also, on our side, service holder agents can secure device because only they have access to network, screen, file system, etc.

Our architecture has four key elements.

- Environment where agents acts – Multi Agent System e.g. Java Based
- Service holder agents
- Information and security agent
- Foreign service agent

*Environment*
Environment must be some Agent based programming environment, where code of this environment is available. Environment has its security manager and certain devices such as screen, network, file system etc. are available only to service holder agents.
In the real life, environment code should be available, signed and proved by different software producers for any security holes. Signatures can be done similar way as signatures of active X controls or other software. A user can check this way that Environment is all right, but except of this he/she can see the source code and look for possible security holes and Trojan horse itself.

*Service Holder Agents (SHA)*
holds its service. They communicate with Information and Security Agent (ISA) only and if ISA passes them foreign service agent (FSA) identifier, they can communicate directly with FSA. ISA passes also allowed communication data (protocol) to SHA. Thus FSA cannot misuse SHA. SHAs can communicate between themselves but always through ISA.

*Information and Security Agent (ISA)*
ISA is designed to communicate with the outside world, with a user and also Foreign Service Agents (FSA). FSAs can use SHAs only through ISA.

*Foreign Service Agent (FSA)*
FSA is service brought by network or whatever connection into environment.
FSA can be brought by different ways

- If a user wants to use some service, he/she makes request to ISA. ISA contacts FSA on some other device on the network. Then FSA comes into environment by Network Service Holder Agent and FSA can start communication with ISA.

- If FSA wants to come to environment, it connects to it and starts communicating with ISA. ISA puts it in a queue, refuses FSA or starts communication with FSA.

FSA is destroyed after all or its incoming instance can migrate into other device.

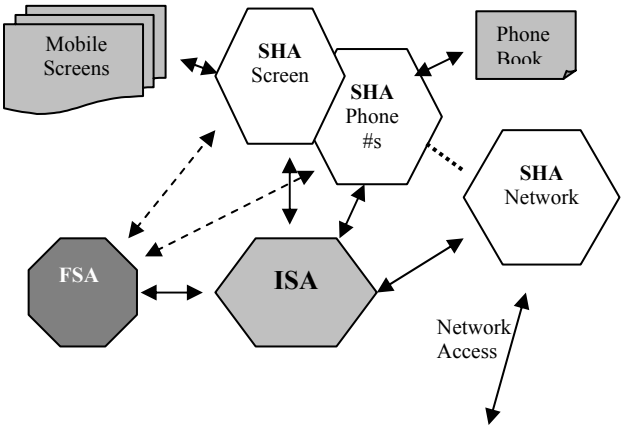


Figure 2 - Architecture

## 4. IMPLEMENTATION AND EXPERIMENTS

Over 80% of recent MAS systems are based on Java that is the most appropriate language for Agents. We build our experiments on IBM Aglets [6] agent system and IBM JKQML class. However, the research and results will be useful for any other Java based agent system. [3]

The following reasons justify Java Programming Language being ideal programming language suitable for our purposes:

- Java supports secure migration of classes.
- Interfaces to systems "speaking" KQML [4] are available for Java.
- Java is platform independent.
- SQL interfaces to databases are implemented.

There are development environments available for Java such as Borland JBuilder, Visual Café etc.

It means implementation of our proposal for Aglets will be usable with little modifications in all other Java Based Agent System.

*Why Do We Use Aglets?*

Aglets were originally created by IBM Japan. Currently it is under GPL (general public license) as Open Source Aglets.org project. Aglets are multi-agent system. An aglet [6] [2] is a Java object that can move from one host on the Internet to another. An aglet that executes on one host can suddenly halt execution, dispatch to a remote host, and resume execution there. When the aglet moves, it takes along its program code as well as its state (data). A built-in security mechanism makes it safe for a computer to host entrusted aglets. Aglets are still under development, which is promising for implementing new features. We decided to use aglets for our experiments. Aglets do not support KQML but using IBM JKQML class we can solve this. [8] Aglets are based on Java that is also very suitable for us.

We focused mainly on wireless systems but we are doing these experiments on regular PC's.

- Wireless connection is replaced by regular network based on IP protocol
- Screen of wireless device is replaced by Java window
- Keyboard can replace device keyboard
- Mouse can replace some pointing device.
- All other conditions are almost same as it would be on wireless devices.

Classes

All implementation what we did so far are just very limited examples. All our classes are extended of Aglet class and they support basic features of secure communication based on RSA and KQML. As an environment we use Aglets with well-defined security manager.

Our security manager is not yet defined perfect but we are working on it. Our goal is not to create functional commercial platform but to show functionality of our proposal on existing MAS with some simple implementation.

| | |
|---|---|
| SHA | Service Holder Agent extended class Aglet. It supports basic features of secure communication based on RSA and KQML. It supports communication with Information and Security Agent. Also basic methods for communication with environment are defined for overwriting. Such as write, read etc. So far it supports very basic features. SHA consists of public key to ISA. |
| ISA | Information and Security Agent extend class Aglet. Communication with SHAs and FSAs. |
| FSA | Foreign Service Agent. It just supports migrating, which is limited by ISA and has knowledge of the same communication protocol as ISA – KQML is used in this case. Also it has method returning identification number of current ISA. |

## 5. EXAMPLES CLOSE TO REALITY

*Sending Postcards*

When we want to send a postcard by some website, we have to write our data such as name, some wishes and also an email address of a receiver. Now let us bring this application into our architecture. ISA makes request to certain FSA, which provides service of the Internet postcard. FSA migrates into Environment of our device. It communicates over ISA and SHA of display with user. A user chooses a postcard and writes text and destination of postcard. FSA creates an FSA, which holds created

postcard in html for example, and this new FSA can be signed by signature SHA with user signature. New FSA is sent to its destination and original FSA is destroyed. Now it depends on destination user if he/she accepts FSA with postcard and views it. After viewing this FSA is destroyed.
This way neither original nor postcard FSA can pass any data to the third party. On the other hand original FSA can view some commercial on user screens. This way sending of postcard means benefit for FSA provider.

*Buying a Book*
This is a bit different example but we will try to explain it by our architecture.
Some agent classes can be stored or deactivated in our device. When we wanted to buy something in the past, we looked on Internet for such agent and made sure it does only what we want. Now we already have buying agent in our device A. FSA (buying agent) is now not foreign but "ours". By ISA and device screen it will ask for a name of a book, a price range and delivery address from the user. We do not want to pass delivery address to device B so we will encrypt it with our private key and public key of post office. FSA leaves device to certain bookstore or starts to search for some stores (it can visit stores from the past for example.). FSA will find the book on device B, negotiate about price, agree or refuse it. If it agrees on the price, FSA will pass encrypted delivery address to ISA on B. This way device B can ship a book to the post office with encrypted delivery address and the post office will know where to ship it but device B cannot misuse our address. ISA on hosted B device can allow agent to send price and payment code back to device A. ISA on A device will activate payment agent and payment agent will make transaction. ISA on B device will check if payment was made. If yes, it will ship the book. If device B did not pass any data except of price to our FSA agent, this can be return back to us with additional information about founded bookstores but it can be also disposed by device B.

## 6. CONCLUSION

When building commercial application, security is the most important issue. We can see how a lot of personal information is misused for different purposes. Solving of those security problems is extremely important. We proposed Secure Agent Architecture for Wireless devices because agents seem to be promising technology to solve these problems. We implemented main parts of our proposal and we described some possible applications for this platform. In our future research we will focus on solving other security problems brought by our proposal such as stealing of Foreign Service Agent by our environment and also we will work on proposing Distributed databases based on our agent architecture.
As we already mentioned, we chose wireless platform because, according to our vision, it is easier to bring new technology into wireless world. However, we will work on device independent architecture in the future.

## REFERENCES


[1] Certification Authority – http://www.verisign.com

[2] Lange, D.: Programming Java Mobile Agents with Aglets. Addison-Wesley, 1998. Canada

[3] Balogh, Laclavik, Hluchy, : Model of Negotiation and Decision Support for Goods and Services, ASIS 2000

[4] KQML Website - http://www.cs.umbc.edu/kqml/

[5] FIPA: Foundation for Intelligent Physical Agents Geneva, Switzerland 1997

[6] IBM Aglets - http://www.trl.ibm.co.jp/aglets/

[7] Laclavik, M.: Negotiation and Communication in Agent Systems, 2001

[8] JKQML IBM - http://www.alphaworks.ibm.com/tech/JKQML

[9] Grasshopper - http://www.grasshopper.de/

[10] TCL - http://agent.cs.dartmouth.edu/general/agenttcl.html

[11] Laclavik, M.: Secure Inter-agent Negotiation and Communication, ICETA 2001

Device-to-Cloud Architecture: IoT modules and gateways, platform & SIMs. The real value of IoT is found in the data that connected products can generate. Mobile networks are vital for acquiring product usage information that OEMs can use to offer new services to their customers or remotely troubleshoot and update their products in the field. Thatâ€™s why Sierra Wireless is building the internet of things with hardware, software, and services that work together to securely deliver valuable data you can use to launch your services faster and drive innovation in your industry. Securely build, c This mini learning guide will cover best practices for achieving and maintaining a secure network architecture, discussing several aspects of DMZ security and VLAN security, including outlining VLAN attacks and how to prevent them. How to compartmentalize Wi-Fi traffic with a VLAN Configuring virtual LANs (VLANs) to tag Wi-Fi traffic can create be an effective way to increase greater security for an enterprise wireless network. Learn the difference between wired LAN and wireless network traffic and how to use these VLAN capabilities, found in both wired and wireless devices, to tag and compart Interact with and configure network devices (switches, wireless access points, firewalls, clients) from more than 150 vendors. Scalability. FortiNAC architecture enables effective scaling to multi-site locations and supporting millions of devices. FortiNAC Models and Specifications.Â  No, FortiNACâ€™s architecture enables complete visibility even from remote locations. There are many organizations that deploy FortiNAC in a cloud such as Amazon Web Services (AWS) to provide NAC for their network. Q. What is the upper limit of how many devices FortiNAC can support?Â  FortiNAC can quarantine a device with a suspicious profile for a network administrator to investigate and resolve. Learn more about Fortinet today. Request a Demo.