An Integrated Secure Web Architecture For Protected Mobile Code Distribution

Watermarking, Obfuscation, Encryption, Digital Signatures and Smart Cards Together Strong against Software Piracy

Mehrdad Jalali-Sohi[1], Rigobert Foka[2] , Gaël Hachez[3], Alexander Beitlich[1]

[1] *Fraunhofer Institute for Computer Graphics*
Rundeturmstr.6, 64283 Darmstadt
GERMANY
jalali|abeitlic@igd.fhg.de,
Tel: +49-6151-155532, Fax: +49-6151-155499

[2]*Thomson-CSF Communications*
160 Boulevard de Valmy/BP82
92704 Colombes Cedex
Paris, FRANCE
Rigobert.FOKA@tcc.thomson-csf.com

[3]*Universtié Catholique de Louvain*
Crypto Group
Place du Levant, 3
B-1348 Louvain-la-Neuve
BELGIUM
hachez@dice.ucl.ac.be

Keywords: mobile agent security, electronic commerce, watermarking, obfuscation, electronic right management, copyright protection, software piracy, reverse engineering

Abstract: IPR (Intellectual Property Rights) protection is one of the key elements to be considered in the development of mobile code technologies (applets, agents, etc.) due to the mobile nature of this kind of software and the power of servers. The absence of protection would increase the risk of piracy to such a level that the economy of this sector would be weakened, perhaps even destroyed.

Complementary to the legal provisions (anti-piracy laws), IPR protection is one of the absolute elements in the development of these new markets. In the course of ESPRIT project FILIGRANE (FlexIbLe IPR for Software AGent ReliANcE), we developed an integrated Web architecture and associated security framework and protocol for the trading of mobile code in Internet. The term *mobile code* includes all kinds of mobile Java software (applets and agents and Java beans, cardlets, etc.).

1. INTRODUCTION

Intelligent software agents are a new class of software that act on behalf of the user to find and filter information, negotiate for services, easily automate complex tasks, or collaborate with other software agents to solve complex problems. People like the idea of delegating complex tasks to software agents. A mobile intelligent agent [2] can move to different agent servers to be executed to perform some tasks. A mobile agent is completely exposed to the server on which it is executed. This is called *Malicious Host* problem. When moving to an untrustedexecution engine (agent server, browser, etc.), casual fraud is of great concern. Copying or reverse engineering the code of the agent is a huge potential threat to the originators of the agents. Mobile agents are an abstraction of mobile code technology by which a program moves between different network nodes. A good overview of different abstractions of mobile code can be found in [7]. During the course of the FILIGRANE project, we developed an integrated security architecture and the associated framework and protocols for mobile code commerce on the Internet. FILIGRANE combines a number of different security blocks to a functioning framework and defines an integrated secure Web architecture for it, addressing the IPR protection of the moving Java code. FILIGRANE is the first attempt world-wide to consider all aspects of mobile code distribution, from production to secure execution on the client platform.

The rest of the paper is organized as follows. Section 2 provides an overview of the FILIGRANE framework and different security blocks. Section 3 presents the Web architecture of FILIGRANE. Section 4 presents an application of an agent-based Visual Mining architecture protected by FILIGRANE solutions. Section 5 presents related work. Concluding remarks and a description of future work are given in section 6.

2. FILIGRANE (FLEXIBLE IPR FOR SOFTWARE AGENT RELIANCE) FRAMEWORK

In this chapter, we describe FILIGRANE security framework and the protocol developed in this project. FILIGRANE offers a complete framework to the software community and addresses security aspects of distribution of mobile code, such as registration of mobile code, certification of the entities involved in an e-commerce scenario, copyright protection, and protection against modifications and malicious use. We use ERMS (Electronic Right Management System), which manages all

kinds of rights and contract handling associated with a piece of mobile code on the provider and the end user platform.

In general, FILIGRANE should cover the following relevant security aspects:

- IPR protection guaranteeing right holders an authorized use of their software,
- authentication mechanisms to identify the mobile code origin and the user,
- security mechanisms certifying the software integrity,
- security protocol for the software download from server to user support,
- mechanisms to trace the software during its entire life cycle, from development to its use by the authorized users,
- mechanisms to protect the end user and executions engine of the mobile code against malicious code behaviour.

A secure software distribution framework should at least support the following points:

- **Secured conditional download:** four main aspects of secure e-commerce over the Internet should be supported:
  - *authentication:* It should be guaranteed that only registered users can download commercial software from the server after successful authentication.
  - *Confidentiality:* It should be guaranteed that nobody can intercept the communication channel.
  - *Integrity:* It should be guaranteed that the software cannot be modified during the transfer.
  - *Nonrepudiation:* It should be guaranteed that the transaction is not repudiable.
  - *Payment:* The payment flow between the entities should be regulated.
- **Right Management:** It should be possible to define and choose conditions for the software. There must be some layer available which manages the rights for different entities based on these conditions.
- **Execution Control:** Execution of the downloaded software should be done in a controlled execution engine. This engine limits access to the software using the Right Management system.

2.1 FILIGRANE Functional Model

The FILIGRANE system is composed of the following actors (see Figure 1) :

- **Certificate Authority (CA)** : a Trusted Third Party (TTP), providing services for the creation and distribution of electronic certificates for the Producer, End User, Provider and the Rights Clearing House (RCH);
- **Producer** : a software developer or company, offering mobile code to a Provider for E-Commerce;
- **Provider** : actor providing goods, such as software, services or information. Provider sells services and/or electronic delivery of items for sale, such as software. Provider negotiates the contract conditions for the use of services electronically;
- **End User** : an authorized holder of a certificate supported by a CA, and registered to perform software downloads by the FILIGRANE system;
- **Rights Clearing House** : this actor is an extension of the IMPRIMATUR [9] IPR Database. The FILIGRANE Rights Clearing House is dedicated to the definition and redistribution of rights between actors of the system as the result of a transaction;
- **Fee Collecting Agency** : this actor is responsible for collecting funds as the result of financial transactions and for redistributing them proportionally to the various actors of the system according to the conditions of the associated contracts. This operation requires a tight linkage between the Fee Collecting Agency and the Rights Clearing House;
- **Quality Label Service** : this optional actor can enter the system with the role of qualifying mobile code to be distributed with various quality labels recognized by potential purchasers;
- **E-Notary** : this actor will notarize all transactions in the system and act as a trusted repository for all actors.

2.2 FILIGRANE Security Blocks

By combining a number of security blocks, we provide a secure framework for the commerce of mobile code over the Internet. These security blocks are presented below.

2.2.1 Object Watermarking

This topic was one of the FILIGRANE research topics. Watermarks were first used in the image / video and audio domain. Extending watermarks to the software domain was a challenging task. The basic requirements or purposes for code watermarks remain the same as for the other domains ([12]). However, a more restrictive constraint is added: a small random modification in the code can break it completely, which is not the case for images or audio. We can translate that requirement in a formal way ([10] extended from [13]):

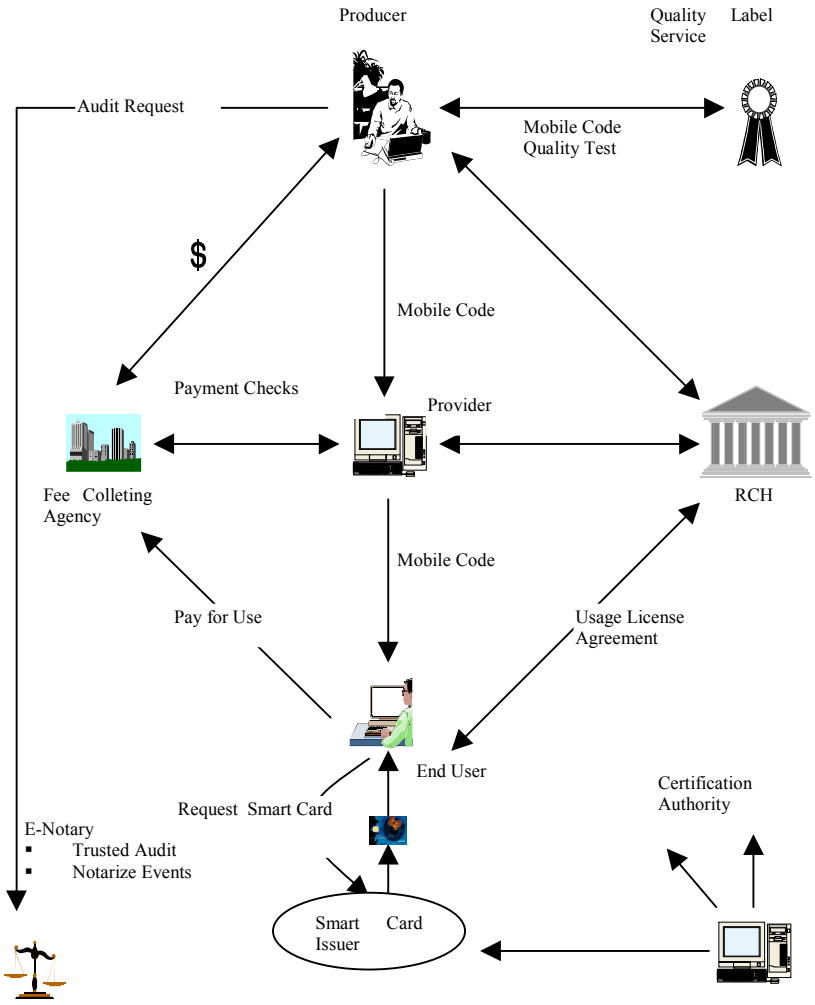

Figure 1 : FILIGRANE Functional Model

Transform a program P into a watermarked program P' with the same observable behaviour.

- If P fails to terminate or terminates with an error, then P' fails to terminate or terminates with an error;
- Otherwise, P' must terminate and produce the same output as P.
- In addition to this constraint, code also adds two new properties that can be used to build better watermarks:
- The watermark modifies the applet (in size, required memory, and/or execution speed) without modifying the user's observable behaviour.
- The watermark can perform a self-check.

Up until now, it seems that three different ways have been explored to implement watermarks in code. The first overview and interesting methods were presented by Collberg and Thomborson in [14],[15] and implemented (Sandmark). They stored some data in a dynamically modified graph structure. The graph structure itself is the watermark. This is done at the source code level, which is a major limitation. A second approach was offered by Venkatesan et al. in [16]. They modified the control flow graph of the program by adding another control flow graph that will be tightly interconnected with the existing one . The authors are not aware of any implementation of this approach.

Both techniques seem to be technically sound, but they do not make use of all the experience acquired in other domains where watermarks have been applied. In [10], we extended an existing robust watermarking technique (spread spectrum) to the code domain. We applied this theoretical scheme to Java code. The main advantages of this technique: it has proved to be robust in the image domain and it can be applied to compiled code (requirement for FILIGRANE). The main limitation: it does not exploit the dynamic behaviour of the code as the two previous schemes; it is a static watermark.

We reuse the spread spectrum technique. This technique adds a mark (constituted by a vector of pseudo-random values with low amplitude) in the frequency domain of the image or the audio signal. It is very difficult to remove and alter this mark (see [17] for more details). The frequency vector where the mark is added is composed by the frequencies of occurrence of doublets and triplets of instructions. In order to add the mark, these frequencies must be modified. The modification is done by building a dictionary of equivalent code instructions. With a heuristic algorithm, code instructions are replaced by an equivalent group of instructions in order to approximate the new frequency vector (the original frequency vector + the mark).

The most difficult part of the implementation is the building of the dictionary of equivalent code instructions. We built a basic dictionary for Java bytecode. As this dictionary is not yet complete, we did not run any extensive tests to assert the robustness of the watermark. We only tested the watermark with the obfuscation module we developed for FILIGRANE (as obfuscation can be seen as an attack on the watermark). The watermark survived and was still readable after the obfuscation. To the authors' knowledge, the robustness of the other schemes was only asserted theoretically and not practically. Once the dictionary is complete and robustness tests have been performed, the watermarking scheme will be presented with full details.

2.2.2 Object Labeling

Two ways for doing so are envisaged: one consists of packaging the Java component (generated bytecode) with a label certifying the code origin and its integrity. This will certify the code origin to the user in order to enhance user confidence in the downloaded code. Technology relies upon classical public keys cryptographic mechanisms, with an adaptation needed for the fact that objects have an active behaviour.

2.2.3 Object Certification

In the FILIGRANE transactional model, we we forsaw Certification of mobile code (Quality Label Service). This consists of attaching cryptographic certificates to the object after the object has been declared ready for it. The object will then be accepted by the execution engine, which will authorize some action by or on this object after checking the certificate. In the certification process, a proving compiler may be used in order to automatically check some characteristics of the agent, based on the source code or some form of intermediate code.

2.2.4 Protection of the Mobile Code Container

Different kinds and levels of protection can be achieved with the mobile code. The FILIGRANE architecture will provide support for some protection mechanisms. Those mechanisms are described below:

- *Signature:* The mobile code could be signed by different parties. This signature can guarantee the origin and the integrity of the mobile code. In addition, each entity can add new signatures as it is required for later checks. The following actors could add signatures to the mobile code container:
  - Producer: Through verification of this signature, the Provider can check the identity of the Producer against database entries for trusted producers.
  - Quality of Label Service: Through verification of this signature, other entities can verify that this piece of code complies with particular levels of quality and conditions.
  - Provider: Through verification of this signature, the end user or other providers could verify the Provider's identity against entries in their database for trusted providers.
- *Encryption:* The encryption of the mobile code has two objectives. The first objective is to avoid a decompilation / reverse engineering. The second is to control parts of the execution / read rights of the customer.
- *Rules:* are associated with the mobile code. These rules will describe some parts of the contract between the producer, the provider and the end user. These rules will be checked by the FILIGRANE execution environment to avoid any breach of the contract.
- *Obfuscation:* Intuitively, the obfuscation of the code has to make modifications in the compiled code in order to make it harder to reverse engineering.

• *Labeling:* A tag is an important piece of data. It permits the identification of the mobile code (name, version, author, date,...). That is why a label must be present with the code. Note also that a part of the (or maybe the whole) tag must be signed to protect the data embedded in it.

Code Envelope:: All of these protection mechanisms must be combined inside a whole package. That is the objective of the code envelope. The code envelope will provide an integrated package where the mobile code and all protection mechanisms associated with this mobile code are stored. We implemented standard PKCS#7 Envelope for Jar files recommended by inventors of the RSA public-key cryptography and matter of standardization efforts by IEEE P1363 program [1].

2.2.5 FILIGRANE Security Engine

Membership to a FILIGRANE Trusted Environment implies:

- Inscription in the Environment's PKI.
- Equipment of systems with the FILIGRANE security engine.

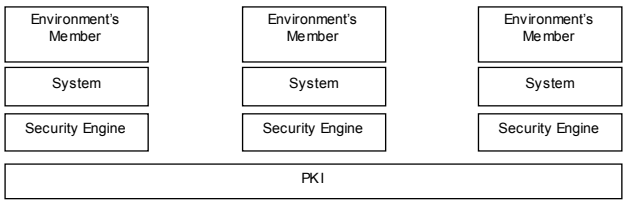

Figure2: FILIGRANE Security Engine

All operations on the mobile code within the FILIGRANE trusted environment for different actors are controlled by a Security Engine. This is the runtime process including all objects classes and services required to facilitate all actions for a particular entity.

The FILIGRANE security engine must be embedded in the mobile code environment of hosts to control mobile code packaging and execution (see Figure2). The execution environment for the client could be a browser, a mobile agent platform or any other kind of mobile code system. The following environment is needed for code execution under FILIGRANE:

- A FILIGRANE security engine must be plugged into the Mobile Code Provider platform in order to control transactions (distribution and deposit of the mobile code).
- A FILIGRANE security engine must be used by the Mobile Code Producer to initially prepare and deposit his code for distribution.
- A FILIGRANE security engine must be implemented at the end user execution environment.

We can imagine that, in the future, providers may wish to install different components into the FILIGRANE security engine (different watermarking algorithms, obfuscation, etc.). Using dynamic class loading of the Java Virtual Machine, we allow the installation of software components to the FILIGRANE security Engine at runtime to provide separate name spaces for various providers.

2.2.6 ERMS (Electronic Rights Management System)

A key part of the FILIGRANE process is the management of rights, expressed in the form of rules attached with the software. This is handled by) the ERMS belonging to the Rights Clearing House (RCH), which will maintain the list of rights associated with a user. The consumer accesses the mobile code through an ERMS (Electronic Rights Management System), which creates a trusted environment between all of the actors of the transaction (user, service provider, content provider). The ERMS can, therefore, be seen as a "logical middleware", as an object "service broker" or perhaps more appropriately as a "trusted intermediation agent" in the transaction/negotiation process.

2.2.6.1 Rules

The rules are conditions that have been agreed to by the communication partners for the exchange of the software. These rules will be checked at the end user's machine before and during execution. The preliminary set of rules identified by FILIGRANE at the present stage is:

- At runtime
  - Time limit: Interval of dates and usage time.
  - Cost: Units/event.
- Before execution
  - On-line check: get authorizations by XYZ.
  - Authentication of code.
  - Versioning / data accuracy.

2.2.7 FILIGRANE Protocol

FILIGRANE Protocol published mainly in [6] is a XML based protocol, especially designed for the secure exchange of information between FILIGRANE Entities. The specification of this protocol mandates an XML vocabulary that is used for representing messages involved and return values. The FILIGRANE protocols enable different actors from the FILIGRANE functional model to a functioning distributed system.

2.2.8 FILIGRANE Launcher

The FILIGRANE security engine for the client is called FILIGRANE Launcher (FL). FL is a runtime process, which creates and manages the interaction between different objects in order to run the mobile code in a controlled environment. For each application, one FL should be started and run continually. Components involved in the secure execution of the software are:

- *Filigrane Launcher (FL):* This is the main process in charge of loading the Filigrane packaged application into memory and running it after successful checks.

- *Rules Manager (RM):* Thread responsible for checking the dynamic rules (runtime execution rules) associated with the mobile code being run.
- *Application Thread:* This thread is running the FILIGRANE downloaded mobile code and is controlled by the FILIGRANE launcher. Its life cycle is controlled by the RulesManager and will be stopped after the counter inside the smart card is empty.

The execution process realizes the control steps on the protected mobile code execution.

3. FILIGRANE WEB ARCHITECTURE

For the integration of FILIGRANE functionality in the World Wide Web (see Figure 3), we implemented a client server architecture composed of only two actors:

- A registered user on the client side who is using a browser to download the mobile code
- The provider on the server side who is running an HTTP server and offers the mobile code to registered users.

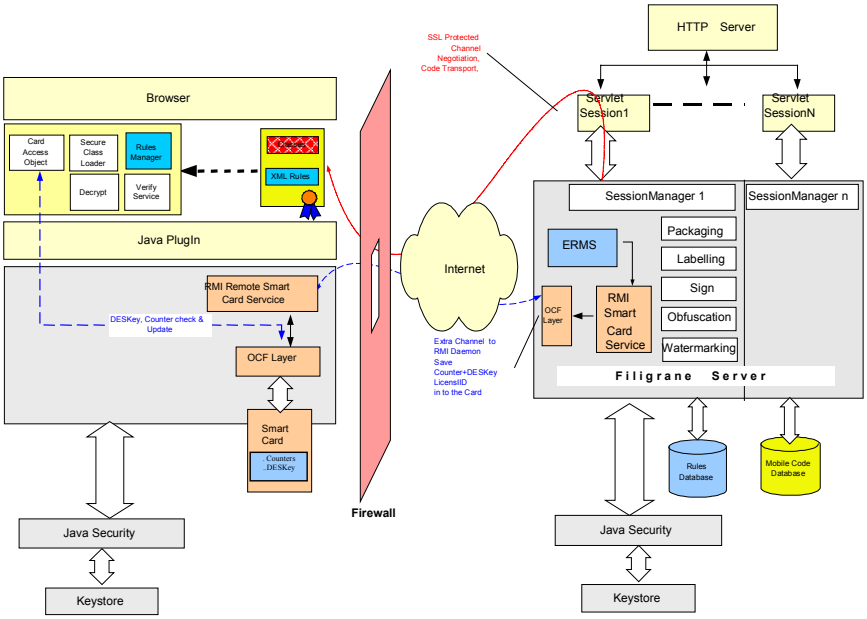

Figure 3: FILIGRANE Web Achitecture

The ERMS system consists of different parts for server and the client. ERMS on the server is the main entry point to the FILIGRANE mobile code server and central entity for right management. ERMS controls all right management questions before the mobile code is transmitted to the client. The counter for the usage of mobile code and the Symmetric Key are sent to the remote card using an extra channel (we use RMI channel at the moment). FILIGRANE Web architecture realizes the following

steps, which are typical for Electronic commerce: Browsing, Item Selection, Ordering, Contract Handling, Authorization, Confirmation, Delivery of Software.

4. AN EXAMPLE APPLICATION USING FILIGRANE

Based on FILIGRANE, we defined an architecture of retail visual mining in Web. In this application, we allow the developers to organize projects and associate particular agents and collections of agents (i.e., agency) over an Internet agent provider protected by the Filigrane solution (see Figure 4). The agent-based visual mining architecture is mainly described in [4],[5]. The visual mining scenario consists of the following main software parts:

- *Agents Community:* these are communities of agents composed of different local and mobile agents.
- *Agent Server:* this is the agent platform that performs secure execution, packaging, sending and receiving of agents. We use the SeMoA [3],[11] agent platform developed by Fraunhofer-IGD, which is designed with a focus on the security requirements of mobile agents.

Filigrane Framework: acts as a secure infrastructure to provide the developers of Visual Mining architecture with appropriate agents over Internet agent providers. The diagram below illustrates this distributed architecture using FILIGRANE.

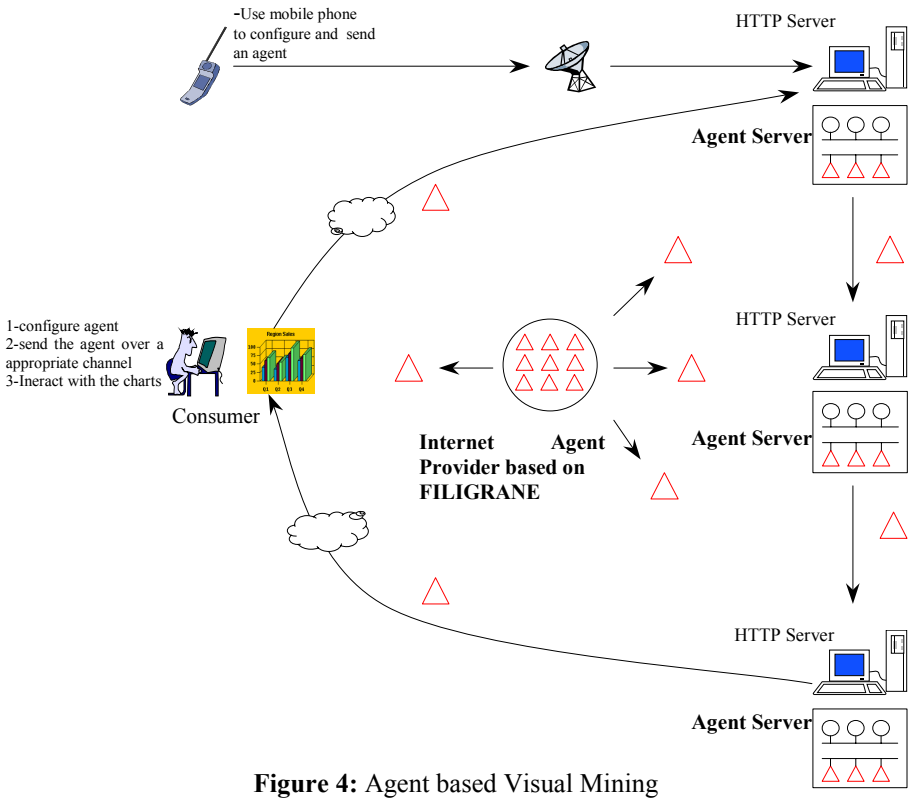

Figure 4: Agent based Visual Mining

5. RELATED WORK

A number of companies, research institutes and organizations are working on concepts, architectures and protocols for the copyright protection of multimedia data. Some projects and developments are successfully attempting to solve this problem for still/video pictures and sounds. At the European level, we can mention the following projects related to FILIGRANE:

- **CITED:** At the European level, the CITED project of ESPRIT has developed a complete model for the management of user rights for interactive multimedia programs.
- **IMPRIMATUR:** This was followed up by the ESPRIT IMPRIMATUR project [9], which aims at ensuring a world-wide consensus on Electronic Copyright Management Systems (ECMS).
- **OCTALIS** [8] This project studied how to couple conditional access and copyright protection on broadband networks (EBU network) and over the Internet.

Close links between the above listed projects and FILIGRANE was developed in order to capitalize on previous developments in a consistent way.

6. CONCLUSION

FILIGRANE addresses all security aspects of dynamic distribution of the mobile code from production to usage and the Monitoring. It performs secure packaging of the content and controlled execution of the software. The FILIGRANE right management system uses a smart card to store the cryptographic usage keys and the usage rights. The content and the rights to use it are and sent separately to the client. It is easy to update the rights without to be forced to download the content again. The smart card in term of security is an important benefit. In the other hand the distribution of the smart card is a practical disadvantage. The smart card is also a portable device, it would be possible to transport usage rights to another device. We believe that the FILIGRANE solution in its architecture has the most key success factors. The FILIGRANE framework has been the focus of protecting intellectual property on the Internet. As a such there are both technological and legislative efforts to continue in order to be successful on the market.

7. ACKNOWLEDGMENTS

This work was performed in the framework of the ESPRIT Project FILIGRANE, funded in part by the European Community under the contract EP28423-FILIGRANE. The authors would like to express their gratitude to the other members of the FILIGRANE Consortium (CSELT, Dados Novos, Euritis, Gemplus) for valuable discussions.

8. REFERENCES

[1] RSA Laboratories, ”Cryptographic message syntax standard,” Public Key – Cryptography Standards 7, RSA Laboratories, Redwood City, CA, USA, 1993. Available at URL: ftp://ftp.rsa.com/pub/pkcs/

[2] Denny Lang, seven good reasons for mobile Agents, and Mitsuru Oshima, Communications of the ACM, March 1999, volume 42, pages 88-89
[3] Volker Roth and Mehrdad Jalali, Concepts and Architecture of a Security-centric Mobile Agent Server, Fifth International Symposium on Autonomous Decentralized Systems, March 26-28, 2001 Dallas, Texas, U.S.A, to appear in IEEE proceedings of the Conference
[4] Mehrdad Jalali-Sohi, Commercial Agent Based Visual Mining in Internet, submitted to the Iternational Conference on Inernet Computing (IC-2001). 25-28 June, Las Vegas, U.S.A.
[5] Mehrdad Jalali-Sohi and Hellene Mamikonyan, An Application of Intelligent Agents for Retail Visual Mining and E-Commerce In Internet, submitted to 14th International Conference COTIM 2001 (Conference On Telecommunications and Information Markets).
[6] Mehrdad Jalali-Sohi, FILIGRANE (FlexIbLe IPR for Software AGent ReliANcE) protocol, A security protocol for trading of mobile code in Internet, 6th International Conference on Information Systems, Analysis and Synthesis (ISAS) 23rd to 26th of July 2000 Orlando, U.S.A
[7] Tod Papaioannou, Doctoral Dissertation, On the Structuring of Distributed Systems: The Argument of Mobility, Loughborough University February 2000
[8] OCTALIS (Offer of Contents through Trusted Access LinkS), http://www.igd.fhg.de/igd-a8/projects/octalis/octalis_en.html
[9] IMPRIMATUR, ESPRIT Project, Project Number 20676, http://www.imprimatur.alcs.co.uk/
[10] Julien P.Stern, Gael Hachez, Fracois Koeune, Jean-Jacques Quisquater, Robust Object Watermarking: Application to Code, In Proceedings of InfoHiding '99, LNCS 1768, pp. 368-378, Sep. 1999
[11] Volker Roth and Mehrdad Jalali, Access Control and Key Management for Mobile Agents, Computer & Graphics, Vol. 22, No. 4, pp. 457-461, 1998
[12] M. Kutter and F. Hartung, Introduction to watermarking techniques. Chapter 5 of S. Katzenbeisser and F. Petitcolas, editors. Information Hiding: techniques for steganography and digital watermarking, 220 pages, Artech House, 2000
[13] C. Collberg, C. Thomborson and D. Low, Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs, ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL98), January 1998
[14] C. Collberg and C. Thomborson, On the limits of Software Watermarking, Technical Report #164, Department of Computer Science, The University of Auckland, August 1998
[15] C. Collberg and C. Thomborson, Software Watermarking: Models and Dynamic Embeddings, ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL99), January 1999
[16] R. Venkatesan, V. Vazirani and S. Sinha. A Graph Theoretic Approach to Software Watermarking. DIMACS Workshop on Management of Digital Intellectual Property. April 2000.
[17] N. Johnson and S. Katzenbeisser. A survey of steganographic techniques. Chapter 3 of S. Katzenbeisser and F. Petitcolas, editors. Information Hiding: techniques for steganography and digital watermarking. 220 pages. Artech House. 2000