SmartContent: A self-protecting and context-aware active content

Akla-Esso Tchao, Giovanna Di Marzo Serugendo
CUI - Université de Genève
Carouge, Switzerland
{*akla-esso.tchao, giovanna.dimarzo*}*@unige.ch*

***Abstract*—The development of communications systems in general, and the Internet in particular, has given billions of people the opportunity to connect and share content with audiences to which they would otherwise never have had access to. Nowadays, anyone can publish and share content, whether personal or not, on the Internet. In addition, the ubiquitousness of mobile devices makes it possible to access content anywhere, at anytime on different platforms. All of this often leads to situations of potential intentional or unintentional misuse of contents as well as privacy problems. Traditional solutions for these problems such as Digital Rights Management have proven not to be appropriate because they rely heavily on costly and centralized external systems or infrastructure. In this paper, we propose *SmartContent*, a novel approach for content protection and privacy. SmartContent acts autonomously and embeds with the content the notion of context and policy. This article presents the general model of SmartContent and an example implementation.**

***Keywords*-self-protection; context; adaptability; policy; negative selection**

I. Introduction

Long ago, mainframe computers were run by experts behind closed doors [13]. Documents leaving the mainframe would likely be printed on paper, or maybe stored on a magnetic tape, and both were controlled physically. Today, almost everyone daily carries one or more personal computing devices with similar or even more advanced performances than the mainframe computers. Moreover, with the evolution of the communication systems, it has become easier than ever to publish and share content, whether personal or not, over the Internet or among personal devices, with the drawback that propagation and copies of this content is hard to control.

Traditionally, Digital Rights Management (DRM) systems are used to protect content (e.g. video, music, computer software, personal information, etc.) against unauthorized access and redistribution. However, these solutions are costly and rely heavily on centralized external systems or infrastructure. They lack the flexibility needed with the development of new technologies. Some example scenarios include the legitimate access of content in unauthorized situations or the legitimate access on any device. For example, a bank employee may not be allowed to access a customer's files from a corporate laptop at home, in a foreign country, or when meeting with other clients, but he should be able to access them on any device within the corporate perimeters. This simple example implies exploiting the agenda of the employee and of the clients, and the location of the employee in order to make decision to grant or not access to the content. This kind of access control over the content can be difficultly achieved with existing solutions.

In this paper we propose *SmartContent*, a *self-protecting* and context-aware active content that can act autonomously and protect itself against any unauthorized or unusual activity. This work is based on our previous work [12] where we introduced the general notion of self-protecting content in the case of personal digital rights management.

The main contributions of this paper are:

(i) A general model for self-protecting content: We propose a model that is as generic as possible in order to enable content protection while ensuring privacy, adaptability and flexibility. Our model can be applied to different scenarios of content protection.

(ii) A context-based content protection: Using the location as context, we propose a first implementation of a location-based content protection that relies solely on the SmartContent, without any external infrastructure or system.

(iii) Multi-levels content protection: A first level of protection is achieved through the usage of standard cryptographic algorithms, a second level of protection through existing obfuscation techniques, and finally a last level of protection through adaptability of SmartContent, both for controlled and uncontrolled environment. For uncontrolled environment (no ties with the content's owner), we propose the *negative selection* mechanism of the *Artificial Immune System (AIS)* as one way to achieve more adaptability.

The rest of the paper is organized as follows: Section II reviews previous and current efforts on self-protecting content. Section III presents the SmartContent model. Section IV shows an example implementation of SmartContent in the case of a location-based content protection. Section V offers a summary of our work and highlights future work.

II. Related Works

To the best of our knowledge, there are very few works related to self-protecting content. Therefore, we will review some of the existing solutions that use a self-protecting content approach or context information to grant access to the content.

The self-protecting document (SPD) [8], [7] is a document that protects itself from uncontrolled use and redistribution. It is comprised of an encrypted document as well as a secure set of permissions and the software necessary to process the document. SPD uses a secure polarization engine packaged with the content to act upon the document before it is stored or decrypted, so the document is never stored in the clear on the user's system. The polarization key used in the decryption and encryption is a combination of data elements taken from the user's system, such as elapsed time since the last keystroke, the processor's speed, and the serial number. Compared to SmartContent, SPD is highly related to the system or device on which it is running on since the generated polarization key depends on the system or device characteristics. Moreover, SPD does not include any notion of context or policy.

The self-protecting container technology (DigiBox) [9] provides a secure container to package information so that the information cannot be used except as provided by the rules and controls associated with the content. It uses encryption, digital signature, and digital certificates to ensure the confidentiality and integrity of the data. Contrary to SmartContent, DigiBox requires a prior deployment of tamper-resistant hardware. In addition, no context information is attached to DigiBox.

Studet et al. [11] proposed a Mobile User Location-specific Encryption (MULE) to encrypt user-specified sensitive files on their laptop. MULE is not self-protecting content but it uses location-specific information from a trusted location to automatically derive a decryption key and allow access to the sensitive files. MULE requires that laptops are equipped with trusted platform modules (TPMs), and a pre-installed Trusted Location Device (TLD). MULE is, therefore, application specific. As opposed to MULE, the context information in SmartContent is more than location. Furthermore, MULE does not embed the protection policy within the content but rather in the laptop. Therefore, an authorized user that extracts the sensitive file from the laptop can access it anywhere.

Finally, Covington et al. [2], introduced the concept of environment role in securing context-aware application. The application is not a self-protecting content but the concept presents how to securely extract, model and use the environment data for access control. However, they consider distributed services, centrally administered using a centralized policy.

III. SmartContent model

SmartContent model shown in Figure 1 consists of two components: the SmartContent and the Renderer. The Renderer may be an external standalone application or embedded within the SmartContent Component.

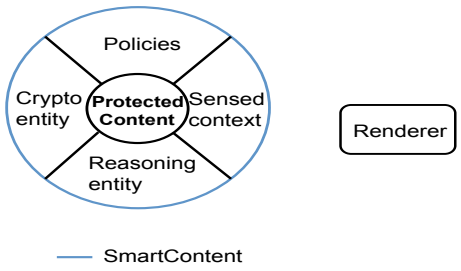

Figure 1. SmartContent's model.

A. The Components

- **SmartContent** - SmartContent is an agent, a piece of software that is embedded with protected content along with policies, the currently sensed contexts, a reasoning entity, and a cryptographic entity. SmartContent acts autonomously on behalf of the content owner for continuous collecting, filtering, processing of information and decision making. It adapts to the changing environment and is capable of continued, autonomous operation while disconnected from the content owner. The different entities of SmartContent are described as follows:
  a) *Protected content* - At the heart of SmartContent sits the content. It is the element we want to protect. It can be an image, a text document, a music file, a movie, a personal information, etc. It is embedded into SmartContent in its protected form (ciphered or scrambled).
  b) *Sensed context* - SmartContent maintains an updated context information within the context entity. This context information can include the current environmental state (location, creation time, day, last update time, platform type, etc.), the content owner identifier, agenda and social connections, law regulations associated with the content at the time of creation and any other context information relevant to the content. The context information is dynamically updated by SmartContent. The context is a set of pairs in the form:
  `Context` :: $\langle\langle$`ctx_name_1`, `ctx_value_1`$\rangle$, $\langle$`ctx_name_2`, `ctx_value_2`$\rangle$, ...$\rangle$
  where: `ctx_name_i` denotes the context name of the pair i and `ctx_value_i` denotes the context value of the pair i. For example:
  $\langle$`gps_position`, `(lat:46.176729, lon:6.139611)`$\rangle$
  represents a context information that indicates the GPS position, `lat:46.176729 , lon:6.139611`.
  c) *Policies* - They are instructions and rules associated with the content. They specify which actions can be performed on the content and in which context. These policies are provided at the creation of the SmartContent or are dynamically acquired. They can be modified and updated according to the situation. A policy is a

tuple of the form:
`Policy :: ⟨action_type, context, properties⟩`
with `properties = (action.property, context.property)`
`action_type` denotes the type of action to perform on the content (read, write, print, etc.), the `context` represents the context information and `properties` denotes the properties associated to the action and the context. For example an action property can be the number of times the content can be read, and the context property can be the acceptable sensed contexts. For example:
`⟨read, ⟨gps_position, pos⟩, (action.property: once, context.property: ctx_property)⟩`
with `ctx_property: pos ∈ [46.1764294, 6.1393119]×[46.1768294, 6.1399119].`
represents a policy that states that the content can be read exactly once, provided `ctx_property` holds for GPS position `pos`.
d) *Reasoning entity* - The reasoning entity is responsible for deciding whether or not to authorize an action on the content. It makes decisions by exploiting the sensed context information, the intended action on the content and the specified policy. Depending on the output of the reasoning entity, the action is or is not authorized on the content.
e) *Cryptographic entity* - It is where all the cryptographic related operations take place. It can encrypt and decrypt the content, and it uses standard cryptographic algorithms and functions.

- **The Renderer** - Once the reasoning entity has decided that the action is authorized, the Renderer is responsible of actually applying the action on the content.

B. Interactions between the components and the entities

Figure 2 summarizes the interactions between the different entities and components of SmartContent. First, the content

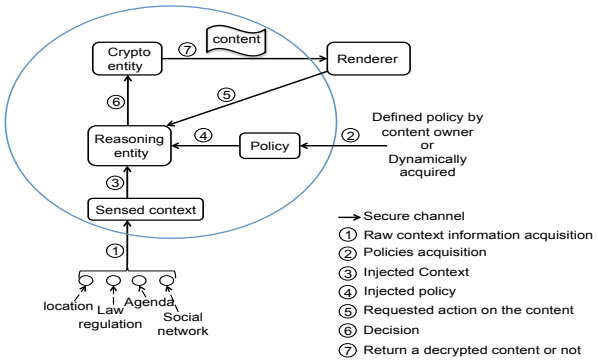

Figure 2. SmartContent's entities and components interactions.

is protected using standard cryptographic functions or algorithms present in the cryptographic entity. These algorithms can be part of a large family of related crypto-algorithms [10] so as to avoid or minimize the break once, break everywhere problem, i.e. when one cryptographic algorithm is broken, few SmartContent instances are affected. Any cryptographic related operation on the content by the cryptographic entity (7), depends on the output of the reasoning entity (6).
To make a decision (6), the reasoning entity exploits: the sensed context information (3) (retrieved from physical sensors in mobile devices or virtual sensors like applications giving social networks information (1)); the action requested by the Renderer (5); and the policy (4) defined or dynamically acquired (2).
The Renderer renders the content if the action is authorized by the reasoning entity.
All the communications and interactions between the different entities are performed through a secure channel, to prevent any leakage of information.

C. SmartContent protection

As any mobile agent, SmartContent (as a whole) can face threats from malicious hosts. Therefore, protecting SmartContent from malicious hosts is similar to protecting any mobile agents or mobile codes from malicious hosts. This problem was extensively addressed in the literature. For the moment, we consider reusing one of the existing techniques: The time-limited blackbox protection through code obfuscation and mess-up algorithms [6]. In this approach, SmartContent code is obfuscated using techniques that are hard to analyze by programs. Since such an obfuscation can be broken by a human attacker given enough time, SmartContent bears an expiration date, after which it becomes invalid. Successful attacks before this expiration date are impossible. Figure 3 summarizes the time-limited blackbox generation process with SmartContent.

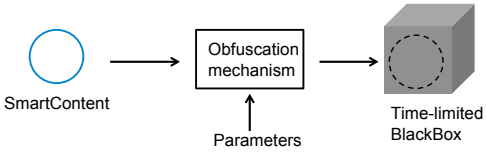

Figure 3. Time-limited blackbox generation process.

In summary, SmartContent protection is achieved at several levels: First the cryptographic entity ensures the protection of the content within the agent (SmartContent), second the time-limited blackbox technique ensures the protection of the whole SmartContent. Moreover, we assume that the sensors, the Renderer, and the SmartContent are securely bound together in such a way that information from a sensor or the Renderer are securely transmitted to the SmartContent (using a secure channel). In general, this applies to any information injected into SmartContent (sensed context,

policy, action). Figure 4 shows the different protection levels in SmartContent.

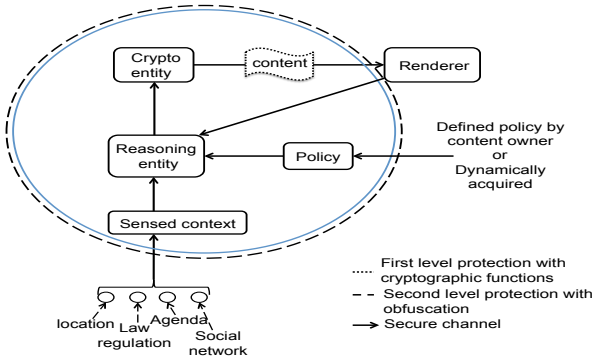

Figure 4. The different protection levels in SmartContent.

Nevertheless, these protections are not always sufficient in particular when SmartContent is let on its own either in controlled or uncontrolled environment. It can be lost, transmitted to unknown recipients, etc. The adaptability of SmartContent plays therefore an important role in the protection of the content in such environment.

D. SmartContent adaptability

The power of SmartContent resides in its ability to adapt and evolve to reflect changes in the environment, without any specific instruction from the content owner. This feature is important since once created, SmartContent is left on its own in the hosting environment. It needs then to learn, adapt and evolve to cope with the changes in the environment. In a controlled environment, this can be done by:

a) Embeding different policies and reasoning algorithms in SmartContent. According to the sensed context, one policy and reasoning algorithm can be triggered instead of another one.

b) Updating and modifying from time to time, the policy rules. SmartContent is modular and hence, any entity in the model can be updated, replaced and modified independently from each other.

In addition to the above adaptability techniques, SmartContent also needs to adapt to unforeseen circumstances particularly inside an uncontrolled environment where ties with the content's owner are severed. Hereafter, we propose one possible mechanism that can be used in such situations: *The negative selection* mechanism of the *Artificial Immune System (AIS)*. Before delving into the details, let us give some short definitions.

The Artificial Immune System (AIS) is an adaptive system inspired by theoretical immunology and observed immune functions and models, aiming at solving problems [3]. *The negative selection* mechanism is one of the metaphors extracted from the human immune system and applied to AIS. This theory is used to explain the ability of the immune system to differentiate between the cells of the organism known as *self* cells, and the foreign elements that can cause disease known as *non-self* cells. The negative selection mechanism is commonly used in research for applications such as virus detection [4], network intrusion detection [5] or hardware fault-tolerant systems [1].

In practice, the negative selection mechanism has two phases, the *Censoring phase* in which the *self* and *non-self* sets are generated and the *monitoring phase* where the detector set (*non-self* set) is put to work. In addition, there is the so called *Co-stimulation* mechanism where a co-stimulation signal is provided by an external entity (human observer for example) to confirm or not, in presence of foreign body, if the latter is *self* or *non-self*. The *non-self* set elements that have detected an anomaly and received confirmation from the external entity, enter a competition and the best of them becomes a memory detector. *Co-stimulation* allows the system to adapt to incomplete or evolving definitions of *self* in the sense that an element in the *self* set can be removed if a negative signal is received from the external entity or promoted memory detector if several positive signals from the external entity are received for that element.

SmartContent can exploit the *negative selection* mechanism in the generation of *non-self* patterns, based on the policy. The *Co-stimulation* can be used for continuous adaptation and evolution of the set of *non-self* patterns inside an uncontrolled environment. In the case the required data for reasoning such as the context is missing or not accessible, depending on the policy and the sensitivity of the content, access can be denied to the content, in order to avoid any risk in such situation.

In the next section, we present an example implementation of the SmartContent in the case of a location based content protection application, followed by a presentation of SmartContent when AIS technique is applied.

IV. Example: A location based content protection using SmartContent

The idea behind the location based content protection is to use the position of a mobile device (smartphone, tablet) as context information, and to authorize or not an action (here "read") on the content on the mobile device.

The position of the mobile device is determined by its GPS coordinates latitude: `lat` and longitude: `lon`.

To achieve this goal, we embed the actual content into a SmartContent and store it in the device.

Figure 5 shows a building map with the SmartContent inside a mobile device. The green zone on the map shows the room where the content can be accessed ("read").

Now, let's see how the different entities of SmartContent described in Section III-A can be implemented for this application.

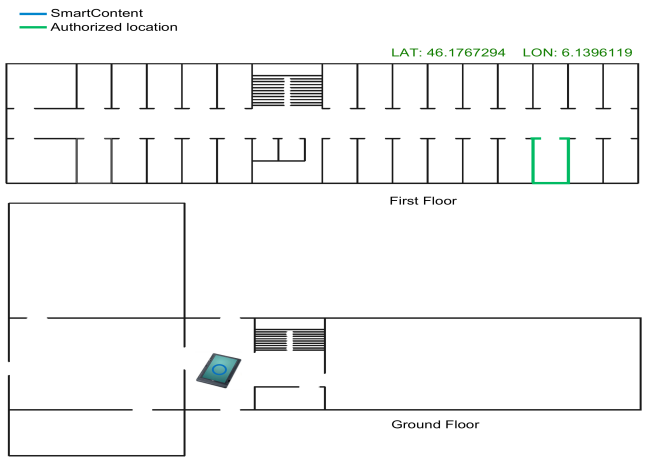

Figure 5. SmartContent's inside the building map.

A. SmartContent

a) *Protected content* - The content is protected using state-of-the art cryptographic algorithms (TDES, AES, etc...).
b) *Sensed context* - Here, the context information is the location (GPS coordinates: `lat` , `lon`). This context information is provided by the mobile device to the SmartContent. Let denote `L` the location. The context information can be represented as follow:

$$\texttt{Context} :: \langle \texttt{gps\_position}, \texttt{L} \rangle \quad (1)$$

c) *Policies* - The policies P in this application are for example:

$$\texttt{Policy1} :: \langle \texttt{read}, \texttt{context}, (\texttt{action.property:multiple}, \texttt{context.property:ctx\_property}) \rangle \quad (2)$$

$$\texttt{Policy2} :: \langle \texttt{print}, \texttt{context}, \texttt{action.property:once}, \texttt{context.property:ctx\_property}) \rangle \quad (3)$$

where `context` in (2) and (3) is given by equation (1).
With `Policy1` the user can read the content at location `L` many times, while with `Policy2` the user can print the content only once at location `L`.
Let $\texttt{lat}_{\texttt{min}}$, $\texttt{lon}_{\texttt{min}}$ denote the minimum latitude and longitude respectively and $\texttt{lat}_{\texttt{max}}$, $\texttt{lon}_{\texttt{max}}$ the maximum latitude and longitude respectively of the room where the content can be accessed. In both cases, the sensed context should satisfy the property `ctx_property`: $\texttt{L} \in [\texttt{lat}_{\texttt{min}}, \texttt{lat}_{\texttt{max}}] \times [\texttt{lon}_{\texttt{min}}, \texttt{lon}_{\texttt{max}}]$.
d) *Reasoning entity* - Anytime a user wants to access the protected content, the action requested on the content is sent via the Renderer to the SmartContent. If the requested action (A) and the location (L) correspond to the policy (P) defined, the reasoning entity authorizes the action on the content. Figure 6 summarizes the reasoning procedure.
We implemented a simple version of SmartContent in the form of an interactive game to download on mobile devices running the Android operating system.[1] This simulation concentrates on the confrontation of the current GPS position (obtained by moving around a character in the building of Figure 5) and the actual room where a document can be viewed. The policy changes each time the game is restarted, the document agrees to be viewed when the GPS coordinates are inside the perimeter of the correct room.

[1]Game available at http://cui.unige.ch/~tchao/SmartContent/

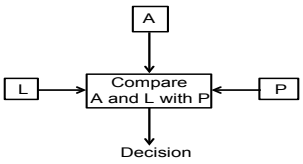

Figure 6. The reasoning process: Action (A), Location (L) and Policy (P).

B. SmartContent with AIS

When using the *negative selection* mechanism described in Section III-D, only the sensed context, the policy and the reasoning entity are modified under AIS.
a) *Sensed context* - With the negative selection algorithm, the context information `L` in equation (1) is encoded in the form of a binary string $\mathbf{L} = \texttt{BSGen(L)}$ using a binary string generator (`BSGen`).
b) *Policies (censoring phase of AIS)* - With the negative selection algorithm, `Policy1`, `Policy2` of equation (2) and (3) are encoded into two sets of binary strings of length l:
$\mathbf{P_1} = \texttt{BSGen(read|L|multiple)}$ and
$\mathbf{P_2} = \texttt{BSGen(print|L|once)}$.
`L` represents authorized locations as specified by `ctx_property`. $\mathbf{P_1}$ and $\mathbf{P_2}$ constitute what we called *self* patterns in Section III-D. From the *self* patterns, the *non-self* patterns can be generated as follows: We place randomly generated strings in a set R_0. The strings in R_0 are tested against the ones in $\mathbf{P_1}$ and $\mathbf{P_2}$. If a string in R_0 doesn't match any binary string in $\mathbf{P_1}$ and $\mathbf{P_2}$, it is considered as *non-self* and added to $\overline{\mathbf{P}}$. $\overline{\mathbf{P}}$ contains randomly generated binary strings that do not match any *self* string in $\mathbf{P_1}$ and $\mathbf{P_2}$ up to a certain threshold. Therefore, $\overline{\mathbf{P}}$ *is not* an exhaustive set of all strings which are not in $\mathbf{P_1}$ and $\mathbf{P_2}$. Otherwise this will require a huge number of *non-self* patterns, depending on the length of the *self* patterns. Paper [12] gives more details on *non-self* patterns generation and some security issues that can arise. Figure 7 summarizes the *non-self* patterns generation process.

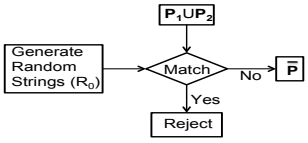

Figure 7. Negative selection algorithm: $\mathbf{P_1}$ and $\mathbf{P_2}$ are binary strings representing the *self* patterns, and $\overline{\mathbf{P}}$ is the set of *non-self* patterns.

In summary, the set of *non-self* patterns $\overline{\mathbf{P}}$ is:
$\overline{\mathbf{P}} = \{$`set of binary strings of length` l / $P_1 \cap \overline{\mathbf{P}} = \varnothing, P_2 \cap \overline{\mathbf{P}} = \varnothing$: `up to a certain threshold` $\}$
c) *Reasoning entity (monitoring phase of AIS)* - With the negative selection algorithm, the Action (A) is also encoded in a binary string $\mathbf{A}$ = `BSGen(A)`. The binary strings $\mathbf{A}$, $\mathbf{L}$ and $\overline{\mathbf{P}}$ are then compared inside the reasoning entity. If $\mathbf{A}$ and $\mathbf{L}$ match up to a certain threshold any string in $\overline{\mathbf{P}}$, the access to the content is denied otherwise, it is granted. It follows the same process as in the Figure 6. In addition, it uses also the co-stimulation signal when needed.

The implementation, using the negative selection mechanism, not only fosters SmartContent adaptivity through co-stimulation signal of AIS (Section III-D), but also provides an additional protection to the content. In fact, the *self* patterns $\mathbf{P_1}$ and $\mathbf{P_2}$ are, in general, hard to detect [4]. Indeed it is, the negative form of the *self* strings ($\overline{\mathbf{P}}$) that is attached to the SmartContent making it hard for a malicious host to guess the authorized location. Even if one instance of the SmartContent is broken, the uniqueness of the *self* string will make it difficult to break all instances.
In addition, since $\overline{\mathbf{P}}$ stores only the negative part of the authorized location, the position of the user's can not be easily deduced from $\overline{\mathbf{P}}$, avoiding therefore any tracking of the user habit by a malicious host.
Figure 8 shows the SmartContent entities with the negative selection mechanism.

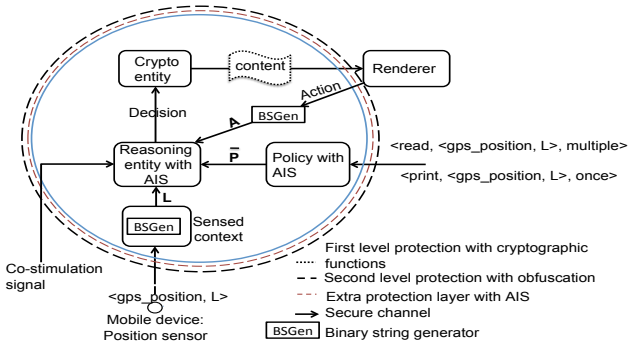

Figure 8. SmartContent, with the negative selection mechanism of AIS. $\mathbf{A}$ is a binary string representing the Action, $\mathbf{L}$ is a binary string representing the location and $\overline{\mathbf{P}}$ represents the *non-self* patterns.

V. Conclusion

In this paper we presented a model to perform a context-aware self-protecting content and showed through an example how this model can be used to achieve a location based content protection. The model proposed is flexible enough and supports adaptability. Moreover, we showed that using additional techniques such as the negative selection mechanism of AIS, we can add more adaptability and an extra protection layer necessary when the content evolves on its own, out of the control of its owner. In our future work, we intend to implement the example presented in this paper on mobile commercial tablets, and to develop an in-door positioning algorithm to retrieve precise GPS positions. We will assess and validate our model in different scenarios and evaluate the protection and adaptability of the content for different use cases. Throughout this work, we will consider light-weight solutions for the implementation.

Acknowledgment

This work has been supported by the EU-FP7-FET Proactive project SAPERE — Self-aware Pervasive Service Ecosystems, under contract no.256873.

References

[1] D. Bradley and A. Tyrrell. A hardware immune system for benchmark state machine error detection. In *Proceedings of the 2002 Congress on Evolutionary Computation, CEC'02*, volume 1, pages 813–818. IEEE Computer Society Press, 2002.

[2] M. Covington, W. Long, S. Srinivasan, A. Dev, M. Ahamad, and G. Abowd. Securing context-aware applications using environment roles. In *Proceedings of the sixth ACM symposium on Access control models and technologies*, pages 10–20. ACM, 2001.

[3] L. De Castro and J. Timmis. *Artificial immune systems: a new computational intelligence approach*. Springer-Verlag, 2002.

[4] S. Forrest, A. Perelson, L. Allen, and R. Cherukuri. Self-nonself discrimination in a computer. In *Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy*, pages 202–212. IEEE Computer Society Press, 1994.

[5] S. Hofmeyr and S. Forrest. Architecture for an artificial immune system. *Evolutionary Computation*, 7(1):1289–1296, 2000.

[6] F. Hohl. Time limited blackbox security: Protecting mobile agents from malicious hosts. *Mobile Agents and Security*, pages 92–113, 1998.

[7] P. Ram, T. Ta, and X. Wang. *Self-protecting documents*. Google Patents, Feb. 2003. US Patent 6,519,700.

[8] P. Ram, T. Ta, and X. Wang. *Self-protecting documents*. Aug. 2005. EP Patent 0,999,488.

[9] O. Sibert, D. Bernstein, and D. Van Wie. Digibox: A self-protecting container for information commerce. In *Proceedings of the first USENIX workshop on electronic commerce*, pages 1–13, 1995.

[10] M. Stamp. Digital rights management: the technology behind the hype. *Journal of Electronic Commerce Research*, 4(3):102–112, 2003.

[11] A. Studer and A. Perrig. Mobile user location-specific encryption (MULE): using your office as your password. In *Proceedings of the third ACM conference on Wireless network security (WiSec'10)*, pages 151–162. ACM, 2010.

[12] A. Tchao, G. Di Marzo, and J.-H. Morin. Personal drm (pdrm) - a self-protecting content approach. In F. Hartung, T. Kalker, and l. Shiguo, editors, *Digital Rights Management: Technology, Standards and Applications*. CRC Press, 2012. (to appear). Available: http://cui.unige.ch/~tchao/papers/pdrm.pdf.

[13] M. Weiser and J. Brown. The coming age of calm technology. In *Beyond calculation*, pages 75–85. Copernicus, New York, NY, USA, 1997.