Self-validating bundles for flexible data access control

P. Gallo, A. Andò and G. Garbo
DEIM - University of Palermo, Viale delle Scienze Ed. 9, 90124 – Palermo, Italy
{pierluigi.gallo},{andrea.ando}@unipa.it

Abstract

Modern cloud-based services offer free or low-costs content sharing with significant advantages for the users but also new issues in privacy and security. To protect sensitive contents (i.e. copyrighted, top secret, personal data, etc.) from the unauthorized access, sophisticated access management systems or/and decryption schemes have been proposed, generally based on trusted applications at client side. These applications, work also as access controllers, verifying specific permissions and restrictions accessing user's resources. We propose secure bundles (S-bundles), which encapsulate a behavioral model (provided as bytecode) to define versatile standalone access controllers and encoding/decoding/signature schemes. S-bundles contain also ciphered contents, data access policies and associated metadata. Unlike current solutions, our approach decouples the access policies from the applications installed in the user's platform. S-bundles are multi-platform, by means of trusted bytecode executors. They offer data protection in case of storage in untrusted or honest-but-curious cloud providers.

1. Introduction

Modern ICT frameworks offer to users several internet-based computing resources accordingly to the Cloud computing paradigm. Cloud applications include e-mail services, data storage, data sharing, social networking, and are consumed by heterogeneous classes of users, through different kinds of devices. On one hand, Cloud computing services deliver several advantages; on the other hand, they introduce new challenges especially on privacy preserving and data security.

In fact, most of cloud applications are based on outsourced-services and rely on. cloud storage. This generally involves cloud providers, beyond the user domain, which have to be considered untrusted, generally honest-but-curious [1,2]. Cloud services include also access control making data accessible only to authorized users. Several methods has been developed to perform access control managements [3], enforced by the user's machine, or by the Cloud. They permit to define specific permissions and restrictions to the user that allow a fine-grain access to resources. On one hand, access control methods that are designed for in-house systems are not suitable for cloud computing applications (as above-mentioned, users and cloud servers are in different trusted domains [1]); on the other hand, cloud-based access control methods rely on trusted could providers. Crypto schemes are used to keep the outsourced sensitive data confidential. However, data will not be available, even to the legitimate user, when he tries to access them from a public device that does not support the specific crypto scheme.

Another privacy issue, related to the data sharing services, is the unauthorized diffusion of sensitive data. When a sensitive content is shared with a limit number of users and, thus, downloaded by one of them (together the decryption key), it can be potentially diffused in a plain form and also consulted in the case of authorization revocation. Digital Rights Management (DRM) [4] systems employ mechanisms that allow preventing the unauthorized diffusion of contents. To get this target, they usually involve proprietary software [5-7] and/or hardware (Apple iPod, Amazon Kindle and Fire). However, Cloud services are offered to several classes of devices, including Personal Computers, smartphones, tablets, smart TVs, etc. Therefore, in order to deploy the above-mentioned DRM mechanisms in different devices, a proper software have to be developed for each platform, with limitations about interoperability and portability. DRM systems also employ in some cases well-known crypto-scheme and access control management that require a performing Internet connection.

Sharing information implies data mobility from users' devices to the cloud, forth and back. Furthermore, multiple copies of these data can be duplicated on multiple devices. Unlike existing access control mechanisms, our solution does not depend on crypto/access control capabilities implemented on the device or on the cloud platform. Our novel methodology associates cryptographic methods and access control rules to data. Our access policies move with data and work also on untrusted devices without specific hw/sw prerequisites. The only requirement is about a multi-purpose execution environment (such as a Java Virtual Machine), which is currently available for a large set of devices.

We therefore propose to enclose data into intelligent security boxes, named bundles. These are indivisible entities that contain data, metadata, access rules and access logic. Using bundles, the access logic and metadata move with data as a second skin, and therefore are available no matter the used device. Multiple copies of the data will include all the relative metadata and access policy.

In our typical scheme, the sensitive data file is encapsulated, in a ciphered form, into a unique bundle that is also Self-validating (S-bundle). Apart from ciphered data, S-bundle contains the access policies and the needed metadata. In Cloud storage environments the S-bundle can be stored in the outsource provider in place of a simple ciphered data file. Thus, when a user downloads a shared file, an S-bundle, that contains the requested content, is provided. Once the user executes it, S-bundle is able to verify the user credentials (that can be composed of a set of keys) and, in the case of positive verification, to allow user to access to data.

With this scheme, the data stored in cloud servers are continuously protected from the unauthorized access. In fact, the sensitive contents are encapsulated into the S-bundle that uses a logical core to perform the access management. Moreover, the data confidentiality is assured as the data owner stores her/his sensitive data into the S-bundle in an encrypted form. Thanks to the S-bundle, data remain protected even if they have been download and stored in untrusted user platforms. The S-bundle in fact continues to provide the access control to data whatever is the environment in which it is executed.

The S-bundle solution fits the needs of cloud-based services and of digital rights management systems (DRM), which are used to hinder piracy and the uncontrolled diffusion of copyrighted contents. As a side effect, S-bundles also guarantee the "right to be forgotten" [8,9] making personal sensitive data no longer available when they are no longer needed for legitimate purposes [10].

The paper is organized as follows: section 2 presents related works while section 3 introduces the self-validation approach, providing details about the S-bundle workflow. Section 4 provides a description of the S-bundle architecture and the procedure to build an S-bundle (S-bundle workflow) are described. Section 5 describes an implementation of self-validation system. Finally, Section 6 draws conclusions and future work.

2. Related works

2.1 Security in the Cloud

Attribute Based Encryption (ABE) is a modern approach to guarantee privacy, security and access control in new ICT frameworks [11-14].

ABE schemes are new types of Identity Based Encryption (IBE) [11,15] that use private keys whose capability to decipher the cipher-text depends on attributes assigned to users. New classes of ABE consider also data access policies, which can be included into the user private key [12] or into the cipher-text [13]. In detail, in these ABE schemes the cipher-text and user private keys are issued considering a set of attributes and access policies. Users can decrypt a cipher-text with their key only whether their attributes satisfy the set access policies. With this approach, data are stored in ciphered form into the cloud providers and are accessed only by authorized users with a limited number of distributed keys.

ABE scheme based on Cipher-text Policy (CP-ABE) [13] thus partially fulfills the features offered by our proposed approach in terms of access control to data but protect them until they remain in a ciphered form. Once sensitive data are deciphered, also unauthorized users can access them.

2.2 DRM systems

DRM systems guarantee compliance with the digital license in terms of access and usage rules. DRM prevents the diffusion of the unprotected versions of digital contents. For this reason, DRM systems are used in several commercial scenarios for selling, lending, renting and streaming of copyrighted contents of different types, such as eBook, games, multimedia, etc.. The DRM world is fragmented because of several standards (DTCP, MPEG-21, OMA, Marlin, etc.) [16-19] and property systems (Adobe Content Server, Apple Fairplay, Microsoft DRM, etc.) [21-24]. This has led to several ecosystems as previously mentioned, this limits system interoperability and portability.

2.3 Right to be forgotten

Internet users have to preserve their privacy that can be compromised by the uncontrolled dissemination of personal information, e-mails, pictures, videos. Controlling data access and maintaining a complete awareness about them is very difficult, once data are made available on the web.

A technique to provide the right to be forgotten to user was proposed by Geambasu et al with a system called Vanish [25]. It exploit the global scale peer-to-peer infrastructure and Distributed Hash Tables (DHT) to map the nodes. Briefly, Vanish encrypts the user's data with a key not know by the user and then destroys it locally. Thus, using a Shamir's secret sharing approach [26], Vanish produces several pieces of the key and sprinkles them at random to nodes of the peer-to-peer networks. DHTs map the nodes that has a piece of key and change continuously. The DHTs are stored only in a limit set of nodes.

After a certain time period, several pieces of the key will naturally disappear because of DHT nodes churn or cleanse themselves. When the number of loosed pieces is greater than the complementary (n-k pieces) of the needed minimum number pieces (k) the key cannot be recovered. In this case, user's data cannot be no longer decrypted and therefore they become unavailable.

Unfortunately, Vanish has some vulnerabilities, faced by other systems, which are more robust against malicious attacks, such as SafeVanish [27] and Sedas [28].

Our S-bundles natively support time-based access control by adding an expired date that has to be checked against a trusted network time protocol (NTP) server. In the case of unsatisfied time conditions, the S-bundle will not permit the access to the nested data.

3. Self-validation approach

S-bundles use a self-validation approach: data are able to self-protect themselves from unauthorized access and become inaccessible beyond a specific expired date. A sophisticated protection wrapper for the data includes several components.

In our vision, S-bundles offer a secure and versatile standalone runnable access controller to data. They are able to guarantee protection of sensitive data when they are stored into external server providers or unsecure user environments. S-bundles enclose user's data, crypto-schemes, access policies, and information related to authorization mechanisms. Despite the concept behind S-bundles is technology agnostic, they can be easily implemented using the Java technology. In facts, the Java virtual machine is the generic executor, and the bundles themselves are like jar files, containing both logic (the bytecode), user data and metadata (the Manifest file).

S-bundles are intrinsically multi-platform because they rely on a standardized executor (in case of Java VM, it is likely already installed in the user platform) to be executed.

Moreover, the content of the S-bundle (data, metadata and logic) is digitally signed and the details of the digital signature and digital certificates are also stored inside the bundle.

Figure 1 describes the steps that an authorized user has to carry out to consume the desired contents in the case of the use of S-bundles.

The user requests the desired content to the server provider that answers with the wrapping S-bundle. Once that user has the S-bundle, the data access logic is executed by the virtual machine executor and data are decrypted.

Tamper resistance mechanisms should be adopted in order to certify that the virtual machine executor is trusted and the S-bundle has not been tampered. In order to validate the virtual machine executor, mechanisms derived from trusted computing solutions (un-tampered hardware and remote attestation) can be employed [29, 30, 31]. Other solutions should be oriented on dedicated chips and/or smart cards for the virtual machine executor [32, 33]. The integrity of the S-bundle can be instead evaluated through digital signatures verification.

Therefore, once that the S-bundle is executed, first it will verify that the virtual machine is trusted and its integrity is not compromise.

Afterwards (in the case of positive validation), S-bundle will ask for user credentials (or other access policies) and these are checked against S-bundle metadata. The credentials can be composed of a proper set of keys, certificates, passwords, etc. The user can be obtained needed keys through different exchange channels (i.e. emails, sms, etc.), needed passwords through specific tokens (One Time Password) and needed certificates by proper cards.

Whether the user credentials are valid, the S-bundle grants access to data.

4. S-bundle architecture

The self-validating bundle represents a protection wrapper in which user data, metadata, certificates and a logical core are encapsulated, The S-bundle architecture is depicted in Figure 2, showing four different components: a data folder, a metadata and information (meta-INF) folder, a certificates folder and a logical part.

The data folder contains different types of user data (personal, copyrighted, etc.) and with different formats and extensions (.doc, .pdf, .jpeg, .avi, etc.). The flexibility offered by the S-bundle allows to manage different class of files and to define a proper protection to them according to their intended use. As you can note in the scheme of Figure 2, the data folder is represented in a ciphered form. The encryption of the data folder is essential to guarantee the protection of the user data when the S-bundle is stored in an honest-but-curious (for instance Cloud storage providers) or untrusted (for instance in the case of unauthorized diffusion) platforms. In fact, in this way the data contained into the S-bundle are accessible only with the proper credentials. The decryption of the ciphered data folder is performed by the same S-bundle in the case of an authorized access request. The used crypto schemes that can be employed to cipher the data folder are not specified, as they are considered flexible.

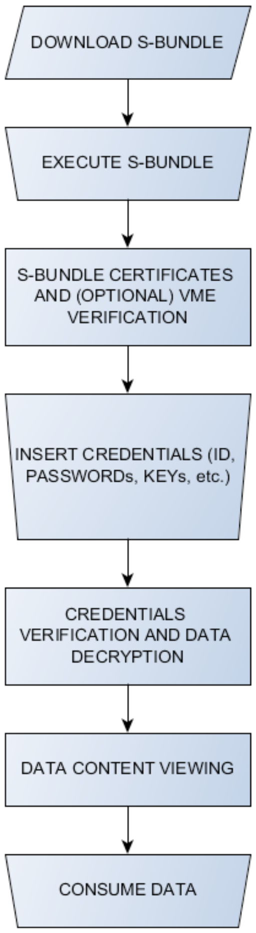
DOWNLOAD S-BUNDLE
EXECUTE S-BUNDLE
S-BUNDLE CERTIFICATES AND (OPTIONAL) VME VERIFICATION
INSERT CREDENTIALS (ID, PASSWORDs, KEYs, etc.)
CREDENTIALS VERIFICATION AND DATA DECRYPTION
DATA CONTENT VIEWING
CONSUME DATA

Fig. 1 - S-bundle consuming flowchart

The data owner is free to choose the crypto scheme that he desires before to generate the S-bundle. Of course, the choice of the crypto scheme involves the contents of other parts of the S-bundle, in particular the meta-INF and the logical part that have to contain the eventual crypto scheme metadata and decryption algorithm, respectively.

The meta-INF folder is contained in the S-bundle in plain form and presents the needed metadata to perform all the S-bundle functionalities. In particular, a file (e.g. the MANIFEST.MF) of the meta-INF folder is dedicated to provide information about the crypto-scheme, the version of the S-bundle, the minimum version of the virtual machine executor, and other useful metadata.

In other files (indicated in the scheme of figure 2 with *first.xml*, *second.xml*, etc.) information about the requested data and the applications that can open them are reported. In detail, they contain descriptions on the user data format, the enabled third-party applications and the attributes of the user data. In particular, this attributes offer details on the operations that an application can carry out on the data files. For instance, whether the attributes offer the only the data file reading, enabled applications will make available only this feature to the user (functionalities of file writing and/or file modifying will be inhibited). However, in order to assure the fair behavior of a third-party application, the latter should be trusted. If an untrusted or malicious application is run by the S-bundle for a data file viewing, users may have more features than how it should have available. In order to overcome this limit and thus to assure the data protection during the content consuming, S-bundles can integrate a trusted data viewer. In this case, functionalities offered by the viewer will follow set data attributes and users will be able to exploit only the proper available features.

The logical part is the core of the S-bundle. In fact, it represents the S-bundle component that allows to accomplish features related to, integrity verification, access control, and content consuming. Thus, the logical part adds active tools to user data making them capable to self-protect themselves from unauthorized accesses and to establish the way in which users consume data files. These features represent an extension of data protection concept that is expressed not only with certificates to assure their integrity (S/MIME protocol with the extensions .p7m, .p7c, p7s and .p10 [34]) but also with a logic that permits to perform an active control (S-bundle) to them. The logical part is made up with compiled codes (optionally obfuscated) and integrates several functionalities. They consist in a friendly Graphic User Interface (GUI), a component for the integrity verification and user credentials validation, a component for the data decryption and, finally, a component for the content viewing (that can be a third-party application or an application integrated into the bundle).

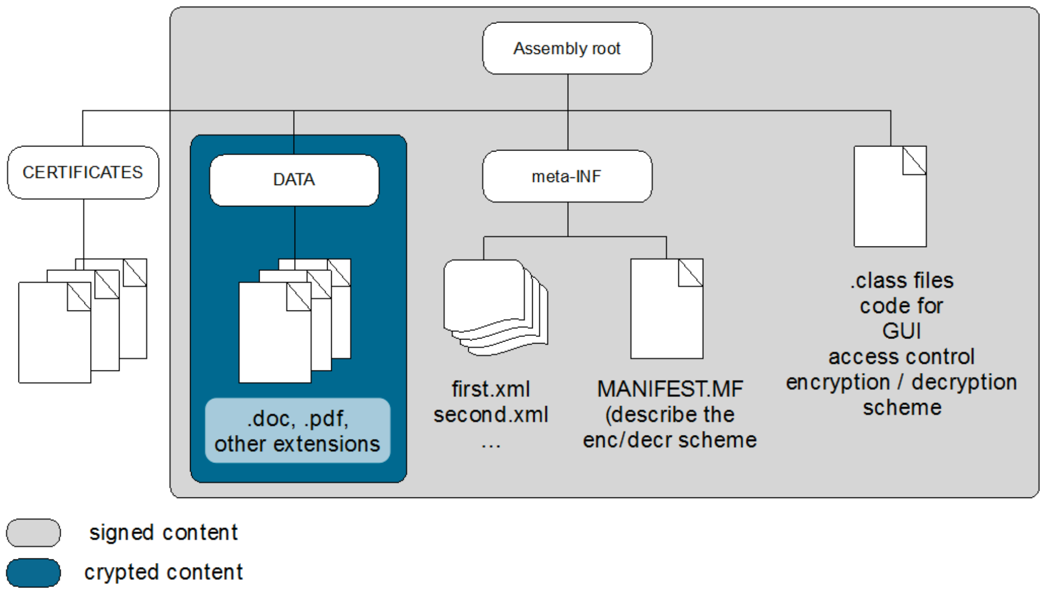
Assembly root
CERTIFICATES
DATA
meta-INF
.doc, .pdf, other extensions
first.xml
second.xml
...
MANIFEST.MF (describe the enc/decr scheme
.class files code for GUI access control encryption / decryption scheme
signed content
crypted content

Fig. 2 - S-bundle architecture

When and S-bundle is executed the first step performed by the logical part is the virtual machine validation. As previously mentioned, it can be performed using mechanisms based on trusted computing that however will not treated in this work. Moreover, the S-bundle carries out the integrity certification. It simply consists on the verification of digital signatures and certificates contained into the S-bundle. Whether the virtual machine executor is not trusted or S-bundle integrity is not confirmed, the logical part advises the user about inconsistencies and thus stops the S-bundle. Conversely, the GUI is launched.

GUI is the first interface that users encounter after the execution of the S-bundle. It has the task of requiring the user credentials. The user will give these information by inserting text into proper text fields (such as IDs, passwords, One Time Password, etc.) or by providing the path for files that contain required certificates, keys and/or other identification data. A verification tool of the S-bundle will thus examine the inserted credentials. In particular, it will verify the identity, the access policies and the keys (inserted by the user). In particular, the keys will be used to perform the decryption process. The mechanisms for the access control procedures depend on the sharing typology, the crypto scheme, the access policy details and the key distribution methods. When user data are shared with only a user, to build the S-bundle is preferable, consider a crypto scheme based on asymmetric cryptography. In this case the data will be encrypted with the public key of the end user. In fact, through the private end user key, the S-bundle will be able to identify the user, to decipher the data and to provide them in a plain form according to the set access policies (that can consider a limited number of access and an expired date).

In the other hand, when the data sharing is oriented to more than one user, mechanism such as One Time Password, online user identification or other suitable solutions should be considered for the S-bundle. For instance, a symmetric cryptography can be used in these cases. However, a distribution key management have to be considered. Moreover, for the user identification other mechanisms should be employed such as an online user authentication (e.g. carried out by an external server with secure protocols).

Novel encryption techniques introduced in Section 2.1 and called Attribute Based Encryption (ABE) [11-14], can be fruitfully used for access control and decryption mechanisms of the S-bundle. In particular, there are ABE classes that are capable to provide a fine-grained access control to resources. We refer to the Key Policy-ABE (KP-ABE) [12] and to the Cypher-text Policy-ABE (CP-ABE) [13]. In these ABE classes, the cipher-text and user private keys are issued considering a set of attributes and an access policy. So that a user can decrypt a cipher-text with her/his key only whether her/his attributes satisfy the set access policy. The attributes and the access policy can be contained into the cipher-text and the private keys, respectively (KP-ABE), or into private keys and cipher-text, respectively (CP-ABE).

The S-bundle flexibility permits to use several alternatives for the access control and encryption schemes, without any pre-defined choice.

The data decryption component of the logical part is composed of computer algorithms that depend on the crypto-scheme used for the data decryption and for the eventual access and identity control crypto-based. Generally, its complexity is related to crypto-scheme. The output of the data decryption component is data in plain form. The latter will be open to viewer tools allowing the user to consume it. As previously mentioned, the viewer tool can be external to the S-bundle or can be integrated in it. Attributes or licenses of use permits to define how the user can consume contents. Of course, in the case of viewer tool integrated into the S-bundle, the rightful use of contents from the user is easier to assure. However, integrating a viewer into the S-bundle can increase a lot its size (this depends on data files format). A third party application excludes this drawback but introduces a security issue related to its trusted behavior. Nevertheless, with a tool validation step (that may be carry out by the S-bundle), also in the case of third party application a fair use of data can be obtained. In order to get the tool validation, the S-bundle have to perform a verification procedure to certificate that the version of the third party tool is trusted. This step can be performed with the aid of an external secure environment consulted by S-bundle, by means the Internet connection and by using secure protocols [29].

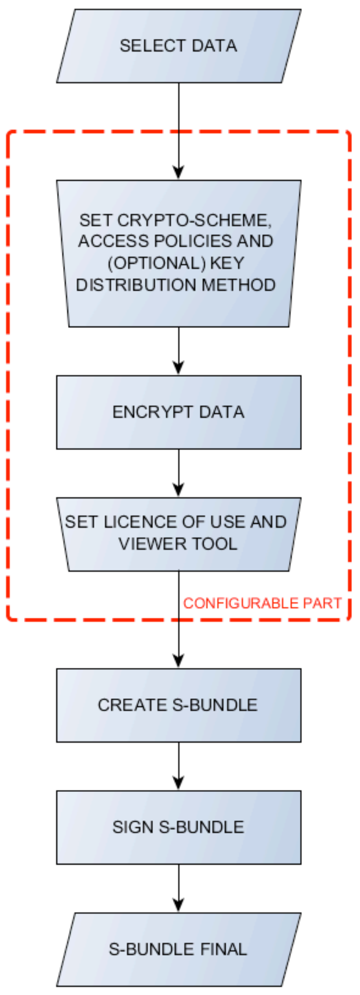
SELECT DATA
SET CRYPTO-SCHEME, ACCESS POLICIES AND (OPTIONAL) KEY DISTRIBUTION METHOD
ENCRYPT DATA
SET LICENCE OF USE AND VIEWER TOOL
CONFIGURABLE PART
CREATE S-BUNDLE
SIGN S-BUNDLE
S-BUNDLE FINAL

Fig. 3 - S-bundle building flowchart

4.1 S-bundle workflow

In the section 3 we have introduced details on steps that a user should carry out in order to be able to consume a content in a self-validation scenario. We have also described that the main element of this approach is the S-bundle. In Section 2 we have described how the S-bundle is composed and what features makes available. According to the specifications introduced in Section 3, we can also present needed steps to generate an S-bundle, starting from data you want share. Figure 3 depicts the flowchart that describes how an S-bundle is build.

A user that wants to encapsulate data into the S-bundle is able to exploit a wide range of configuration settings. In fact, before the building of the final version of the S-bundle (with the signature), the user can select several settings: the crypto scheme, who can access to data (i.e. access policy), how contents have to consume (license of use), what viewer can be used to open data (that can be also integrated into the bundle).

A proper S-bundle builder (S-builder) is provided to the user to aids her/him during the generation steps. Moreover, S-builder is capable to perform other functionalities such as the data encryption and (optionally) the key distribution. In particular, after a user defines the crypto scheme and the access policies, the S-builder will perform data encryption. Once data are encrypted, the user will be able to select the eventual key management. The same S-builder will distribute keys to the end users only after that the S-bundle is signed.

During the procedure to generate the S-bundle a user can also define what third party software will be enable to open user data. Alternatively, a user can also select to integrate a proper viewer into the S-bundle.

5. Our implementation

We have developed a first version of self-validation system that will be under further future improvements and tests. In particular, we have designed our self-validating bundle considering Java as programming language, for the functionalities that it offers in terms of portability, upgradability and for several available libraries privacy and security oriented. Java Virtual Machine (JVM), used to execute java codes, is further widespread among users making most platforms ready to execute S-bundles. The crypto-scheme used for our implementation is the CP-ABE (presented in Section 2.1). This because it has remarkable features related on the fine-grained access control and needs reduced number of keys to be distributed.
In CP-ABE [13] a universe *U* of attributes is considered. Moreover, from *U*, an access structure *A* is also defined in order to set the desired access policy. An authority issues user private keys (UKs) with the function *Setup(PK,MK,U)* where MK is the master key - that is kept hidden - and PK is the public key. Every UK is generated keeping into account attributes associated to the user. For the encryption procedure, that can be performed by each user, an access structure *A* has to be defined. *A* is a tree of nodes derived from desired logic relations (AND, OR, 2of3, etc.) between attributes of *U.* A message *M* can be thus encrypted with the function *Encrypt(PK,M,A)* that outputs the cipher text CT. In order to decrypt CT, the function *Decrypt(PK,CT,UK)* is used from users. In particular, the plain message *M* is outputs from *Decrypt* only if the user attributes (considered into the UK) satisfies the access structure *A*. A java version of previous CP_ABE functions are used in our implementation.
Our S-bundle version is represented by a runnable .jar file in which is contained: a folder with .class files (logical part); a folder with ciphered user data and a folder with metadata (i.e. license of use, CP-ABE public key, etc.), signatures and certificates. In particular, the user certificates has been generated with the "keytool" functionality, made available by Java Runtime Environment. In the same way, in order to sign the .jar file and also to verify it, the "jarsigner" tool has been used. This tool is used to build and to sign S-bundles, and is also to verify S-bundle integrity.
To build s-bundles we have encrypted data with the CP-ABE scheme with a specific number of access structures. We have also set the needed metadata, performed java codes obfuscation, encapsulated and signed S-bundles. The .jar file generated represents our version of S-bundle.
The GUI developed for S-bundles allows users to insert the path of her/his CP-ABE private key. It also provides to users eventual advises derived from errors, security issues, etc. Having the user private key, the S-bundle logical part is able to decrypt data files and then to make them available to user (with limits defined in license of use). A viewer for text files is integrated into S-bundles whereas third party applications (already installed in the user platform) are used for the other classes of files. The S-bundle logical part protects the open files to prevent the user to store a plain version of them.
The JVM validation is another important procedure to accomplished in order avoid possible attempts to compromise S-bundles. Nonetheless, we have chosen to do not developed this module for this first implementation in which JVM is considered trusted. As JVM, third party applications are considered trusted. Ad-hoc mechanisms for JVM and third party applications validation to be include into S-bundles will be investigated and developed in future works. However, the lack of these two modules does not compromise the concept of self-validation approach proposed in this paper.

6. Conclusions

We have presented an access controller based self-validation approach. The data are encapsulated in a wrapper (S-bundle) that is standalone, multi-platform, digitally signed, uses crypto schemes to cipher the data (selected by the user), optionally equipped with a tamper protection. It offers a logical part able to perform the access management, the data decryption and the managing of data consumption. Thanks the above-mentioned logical part, data are able to self-protect themselves from unauthorized accesses. A user that wants to access to data has to execute the S-bundle that with the logical part is able to verify the user credentials (id, password, keys, etc.), to decrypt the data (in the case of positive verification) and also to provide the data contents to the user according to the set license of use. Data can be opened with a third party application or with a viewer integrated into the S-bundle. Software verification procedures are kept into account to validate the virtual machine executor and third party applications in order to assured their trusted behavior.

Our approach represents a flexible and secure alternative to the current commercial solutions in every environment in which the security and privacy of the data is a main topic. Cloud computing and data sharing, electronic Health, Armed Force are just some examples of application scenarios.

Acknowledgment

This work has been partially founded by the national project PON04a2_C SMART HEALTH - CLUSTER OSDH - SMART FSE -STAYWELL, which applies these results to several e-health scenarios.

References

[1] J. Li, X. Chen, J. Li, C. Jia, J. Ma and W. Lou, "Fine-Grained Access Control System based on Outsourced Attribute-based Encryption", in *Proceedings of the 8th European Symposium on Research in Computer Security (ESORICS 2013)*, Egham, UK, Computer Security - ESORICS 2013, Lecture Notes in Computer Science, Springer, Vol. 8134, pp 592-609, 2013.
[2] Y. Zhu, D. Ma, C.J. Hu, and D. Huang, "How to use attribute-based encryption to implement role-based access control in the cloud", in *Proceedings of the international workshop on Security in Cloud Computing 2013 (Cloud Computing '13)*, ACM, pp 33-40, 2013.
[3] P. McDaniel, A. Prakash, *"Methods and limitations of security policy reconciliation"*, ACM Trans. Inf. Syst. Secur., 9(3), 259(291), 2006.
[4] Qiong Liu, Reihaneh Safavi-Naini, and Nicholas Paul Sheppard, "Digital rights management for content distribution", in *Proceedings of the Australasian Information Security Workshop on ACSW Frontiers 2003*, February 2003, Adelaide (Australia), Volume 21, pp. 49-58, 2003.
[5] www.adobe.com/solutions/ebook/content-server.html.
[6] "FairPlay", www.wikipedia.com.
[7] "How FairPlay Works: Apple's iTunes DRM Dilemma", www.roughlydrafted.com, 2007.
[8] R.H. Weber, "The right to be forgotten - More than a Pandora's Box?", *Journal of Intellectual Property, Information Technology and Electronic Commerce Law*, Vol. 2, Iss. 2, 2011
[9] European Commission, "Factsheet on the 'Right to be forgotten' ruling", (C-131/12)
[10] European Commission, "Communication from the commission to the European parliament, the council, the economic and social committee and the committee of the regions", COM(2010) 609 final, Brussels, 2010.
[11] A. Sahai and B. Waters, "Fuzzy identity based encryption", *Advances in Cryptology V EUROCRYPT*, vol. 3494 of LNCS, pp. 457-473, 2005.
[12] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data", in *Proceedings of the 13th ACM conference on Computer and communications security*, pp. 89-98, 2006.
[13] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption", in *Proceedings of IEEE Symposium on Security and Privacy*, pp. 321V334, 2007.
[14] G. Wang, Q. Liu, and J.Wu, "Hierachical attibute-based encryption for fine-grained access control in cloud storage services", in *Proceedings of the 17th ACM conference on Computer and communications security*, pp. 735-737, 2010. [15] A. Shamir, "Identity-Based Cryptosystems and Signature Schemes", in *Proceedings of CRYPTO 84*, Lecture Notes in Computer Science, Springer, Vol. 196, pp 47--53, 1985.
[16] www.dtcp.com
[17] mpeg.chiariglione.org/standards/mpeg-21/mpeg-21.html
[18] openmobilealliance.org
[19] www.marlin-community.com
[20] www.adobe.com/solutions/ebook/content-server.html
[21] en.wikipedia.org/wiki/Adobe_Content_Server
[22] en.wikipedia.org/wiki/FairPlay
[23] en.wikipedia.org/wiki/Windows_Media_DRM
[24] en.wikipedia.org/wiki/Active_Directory_Rights_Manag ement_Services
[25] R. Geambasu, T. Kohno, A. A. Levy, H. M. Levy "Vanish: Increasing Data Privacy with Self-Destructing Data", in *Proceedings of the USENIX Security Symposium*, Montreal, Canada, August 2009.
[26] A. Shamir, "How to share a secret", *Commun. ACM*, 22(11), pp 612–613, 1979.
[27] L. Zeng, Z. Shi, S. Xu, D. Feng, "SafeVanish: An Improved Data Self-Destruction for Protecting Data Privacy", in *Proceeding of the Second International Conference on Cloud Computing Technology and Science (CloudCom), IEEE*, vol., no., pp.521,528, 2010.
[28] L. Zeng, S. Chen, Q. Wei, D. Feng, "SeDas: A Self-Destructing Data System Based on Active Storage Framework,", *IEEE Transactions on Magnetics*, vol.49, no.6, pp.2548, 2554, June 2013.

[29] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum and D. Boneh, “Terra: a virtual machine-based platform for trusted computing”, *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, ACM Press, Bolton Landing, NY, USA, pp. 193-206, 2003.
[30] www.trustedcomputinggroup.org
[31] D. Schellekens, B. Wyseur, B. Preneel, “Remote attestation on legacy operating systems with trusted platform modules”, *Science of Computer Programming*, Vol. 74, Is. 1–2, pp 13-22, 2008.
[32] www.arm.com/products/processors/technologies/jazelle. php
[33] www.oracle.com/technetwork/java/embedded/javacard
[34] S. Dusse, P. Hoffman, B. Ramsdell, L. Lundblade L. Repka, “S/MIME Version 2 Message Specification”, *RFC 2311*, March 1998.