Deliverable No. 10.3

The CHIC encryption services

Grant Agreement No.:	600841
Deliverable No.:	D10.3
Deliverable Name:	The CHIC encryption services
Contractual Submission Date:	31/03/2015
Actual Submission Date:	07/04/2015

Dissemination Level		
PU	Public	
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	**X**

COVER AND CONTROL PAGE OF DOCUMENT	
Project Acronym:	**CHIC**
Project Full Name:	Computational Horizons In Cancer (CHIC): Developing Meta- and Hyper-Multiscale Models and Repositories for In Silico Oncology
Deliverable No.:	D10.3
Document name:	The CHIC encryption services
Nature (R, P, D, O)[1]	O
Dissemination Level (PU, PP, RE, CO)[2]	CO
Version:	1
Actual Submission Date:	07/04/2015
Editor: Institution: E-Mail:	Simone Bnà CINECA simone.bna@cineca.it

ABSTRACT:

This deliverable describes the encryption services, which were integrated to the hypermodelling framework as part of the CHIC infrastructure.

KEYWORD LIST:

Hypermodelling framework, architecture, encryption.

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 600841.

MODIFICATION CONTROL			
Version	**Date**	**Status**	**Author**
1.0	16/03/15	Draft	Simone Bnà, CINECA
2.0	30/03/15	Draft	Simone Bnà, CINECA

[1] **R**=Report, **P**=Prototype, **D**=Demonstrator, **O**=Other

[2] **PU**=Public, **PP**=Restricted to other programme participants (including the Commission Services), **RE**=Restricted to a group specified by the consortium (including the Commission Services), **CO**=Confidential, only for members of the consortium (including the Commission Services)

3.0		Draft	

List of contributors

- Simone Bnà, CINECA
- Elias Neri, Custodix

Contents

1 Executive Summary

The purpose of this document is to describe the encryption services implemented in the CHIC hypermodelling infrastructure. Starting from a brief presentation of the implementation adopted in a previous project called PhysiomeSpace, we discuss the implementation choices of the CHIC encryption services. These involve the development of:

- a module in the hypermodelling infrastructure to encrypt and decrypt data according to the AES-CBC algorithm
- an interface to an external service provided by a member of the consortium where the passphrase of the AES-CBC algorithm is stored

A schema of the several steps of the encryption/decryption algorithms is also presented.

The encryption services are now available as part of CHIC VPH-HF to be used whenever needed in the storing of resources.

2 Introduction

2.1 Purpose of this document

During the second year of the CHIC project, WP7 main goal has been the release of the first prototype of the hypermodelling framework and its testing with respect to the execution of hypomodels and hypermodels provided by the CHIC modelling partners [1]. The hypomodels and hypermodels require input data and produce output results which can contain sensitive personal information or simply a private content that might be read by an unauthorized user.

To enhance the security and protection of data from malicious users, an encryption service has been prescribed to be added to the framework as part of WP10 activities. This document will provide a description of the encryption service in terms of functionalities, APIs and implementation choices.

2.2 Structure of this document

After a brief introduction on which is the role of the encryption in the framework, one section is dedicated to the solution that was adopted in the PhysiomeSpace project (http://www.physiomespace.com) and another section has been added to describe the current implementation of the encryption services inside the CHIC VPH-HF framework.

3 CHIC encryption services

3.1 The role of encryption

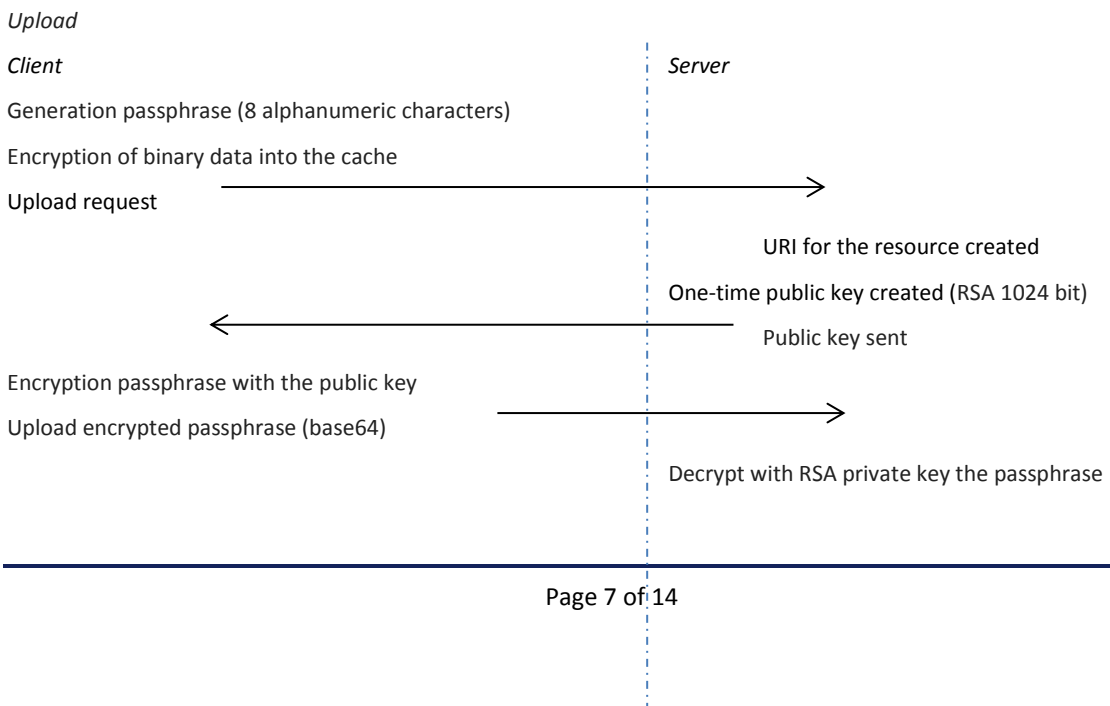
In cryptography, **encryption** is the process of encoding messages or data in such a way that only authorized parties can read the content. Encryption does not itself prevent the interception, but it denies reading the message content to the interceptor. An encryption scheme usually uses a pseudo-random encryption key generated by an algorithm to encrypt the message or data. It is in principle possible to decrypt the message without having the key, but if the encryption scheme is well-designed, the time required becomes prohibitive even if large computational resources are used [2].

From this brief introduction, the importance of storing the key in a secure place emerges. One possible solution (the one we will adopt also in CHIC) is to store the key and the encrypted data on different servers. This solution enhances the security of the framework because the malicious user has to know how to grab the key as well as the encrypted data, and they would also have to have access to both servers.

In the next section we will describe briefly the solution that was adopted in PhysiomeSpace and the solution used in the CHIC-VPH-HF framework.

3.2 The PhysiomeSpace solution

PhysiomeSpace is the digital library service designed to help researchers to share their biomedical data and models [3]. The upload/download services of PhysiomeSpace use a public-key cryptography algorithm to make sure that the transferred data cannot be opened if accessed by unauthorised people [4]. Public-key cryptography refers to a cryptographic system, which requires two separate keys, one to lock or encrypt the plaintext, and one to unlock or decrypt the cypher text. The distinguishing technique used in public key cryptography is the use of asymmetric key algorithms, where the key used to encrypt a message is not the same as the key used to decrypt it. Each user has a pair of cryptographic keys, a public encryption key and a private decryption key. The publicly available encrypting-key is widely distributed, and the private decrypting-key is known only to the recipient. Messages are encrypted with the recipient's public key and can be decrypted only with the corresponding private key. The keys are related mathematically, but parameters are chosen so that determining the private key from the public key is prohibitively expensive. This encryption mechanism has been implemented in PhysiomeSpace using the *pycrypto* library [5]. Figure 1 represents the scheme on how the system works between the client and the server side for each data resource.



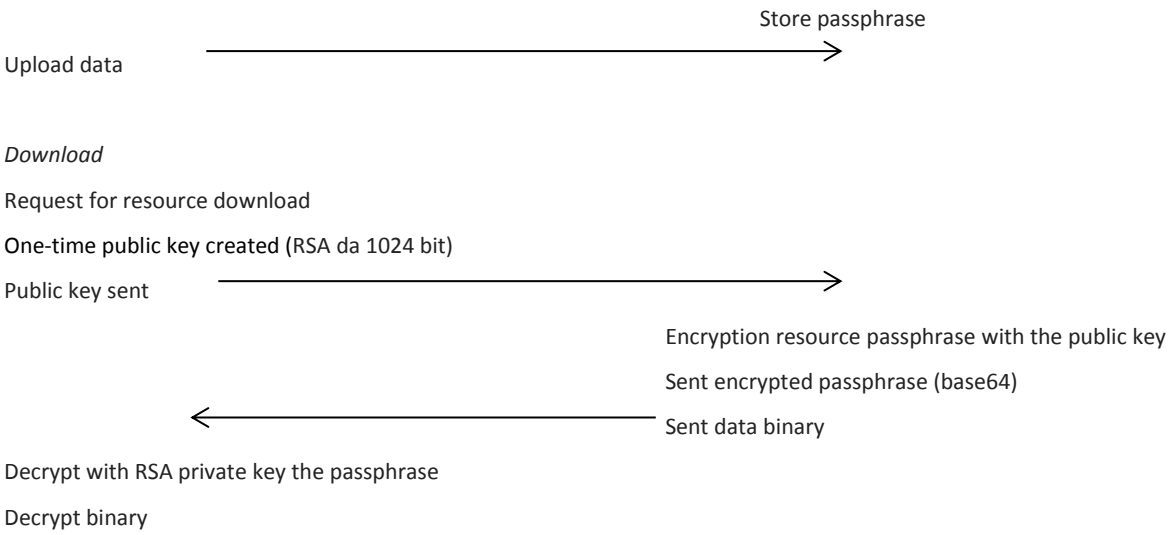


Figure 1 – Schematic representation on the PhysiomeSpace encryption mechanism

The encryption mechanism was also used by the VPH-OP hypermodel framework "template wrapper" to encrypt/decrypt the data when connecting to the storage services.

In terms of security of the data transfers,

- when a workflow is submitted:
  - the Workflow Manager securely stores the user's Session Ticket,
  - the Workflow Manager makes the Session Ticket available to all involved services.
- The Storage Service grants access to the user's resources only if:
  - the given Session Ticket is validated by the Authentication Service,
  - the requestor is registered at the Registry Service,
  - the requestor is involved into the workflow execution.
- Data is encrypted with the previously described asymmetric key algorithm.
- All communications over HTTPS protocol.

3.3 The CHIC encryption services

The necessity to protect data from unauthorised users is a requirement in the CHIC project. The solution adopted in CHIC consists of using the AES-CBC-256 **symmetric-key** algorithm to encrypt the resources. Symmetric-key algorithms are algorithms for cryptography that use the same cryptographic key for both encryption of plaintext and decryption of ciphertext. This implies that the key must be stored in a secure place and preferably on another location then where the encrypted data are stored, as we mentioned in the section above. For this reason, Custodix[3] has provided, as part of the CHIC security framework, a service to store the key.

As described in Deliverable 5.2 the CHIC security framework is build up from different distinct components [7]. Users, their roles and attributes are managed through the Identity Management component (IdM). This component consists of a set of user management pages, REST and SOAP web services. It would therefore be logic to also make the IdM responsible for the management of the

[3] Custodix is a private company member of the consortium of the CHIC project

user encryption keys. For this the IdM was extended with the ability for users to manage their own custom attributes through the REST service "/idm/services/rest/chic/attribute".

This service provides REST APIs to get, store and delete an attribute:

- To get the value of an attribute named {attributeName}

  GET /idm/services/rest/chic/attribute/{attributeName}

- To delete the attribute named {attributeName}

  DELETE /idm/services/rest/chic/attribute/{attributeName}

- To store a new attribute named {attributeName} with as value {attributeValue}

  POST /idm/services/rest/chic/attribute/

  where the payload is

  ```
  <attribute type="xsd:string" name="key"
  xmlns="http://model.webServices.idm.ciam.custodix.com/">
    <value>>{attributeValue}</value>
  </attribute>
  ```

In each request, it is also mandatory to specify the content-type header, which is 'application/xml' and the authorization header for authentication (users are only able to access and modify their own attributes). Only authentication through a SAML token is supported. To create a SAML token you need to perform the following steps:

1. Get the SAML token from the CHIC Secure Token Service
2. ZLIB (RFC 1950) compress the retrieved SAML token
3. Base64 (RFC 4648) encode the compressed SAML token
4. Supply an "Authorization" header with content "SAML auth=" followed by the encoded string:

   Authorization : SAML auth=<Base 64 encoded compressed SAML token>

This attribute service is used within the encryption services by the end user to store their symmetric encryption keys.

The encryption services have been integrated into the VPH-HF framework through the Storage Management System. The Storage Management System (SMS) is the module responsible to retrieve and store data from a storage repository. The SMS can be configured to enable/disable the encryption services according to the needs during the deployment phase; this will allow, in cases where no security issues are present, to disable the encryption and thus have a more efficient communication. The SMS encryption module generates a 256 bit random key for every new file we want to encrypt. After the file is encrypted, the key is stored by making a POST request to the URL specified above. To enhance the security of the system, the key is encrypted and 64-based encoded before being uploaded to the Custodix web server. We use an RSA public key encryption protocol according to PKCS#1 OAEP (RFC 3447) [6]. This scheme is more properly called RSAES-OAEP [5].

As explained in the previous section, the RSA public-key cryptography refers to an asymmetric cryptographic system, which requires two separate keys, one to lock or encrypt the plaintext, and one to unlock or decrypt the cypher text. When the VPH-HF framework is deployed on a new machine server, a public-encryption and a private-decryption key pair is generated. The private key must be stored in the server in a secure place in order to forbid a malicious user from stealing it. The AES-key is encrypted with the public RSA key before being uploaded to the Custodix web server and

decrypted after being downloaded. A schema of the encryption/decryption algorithm is shown in the next page (Figure 2).

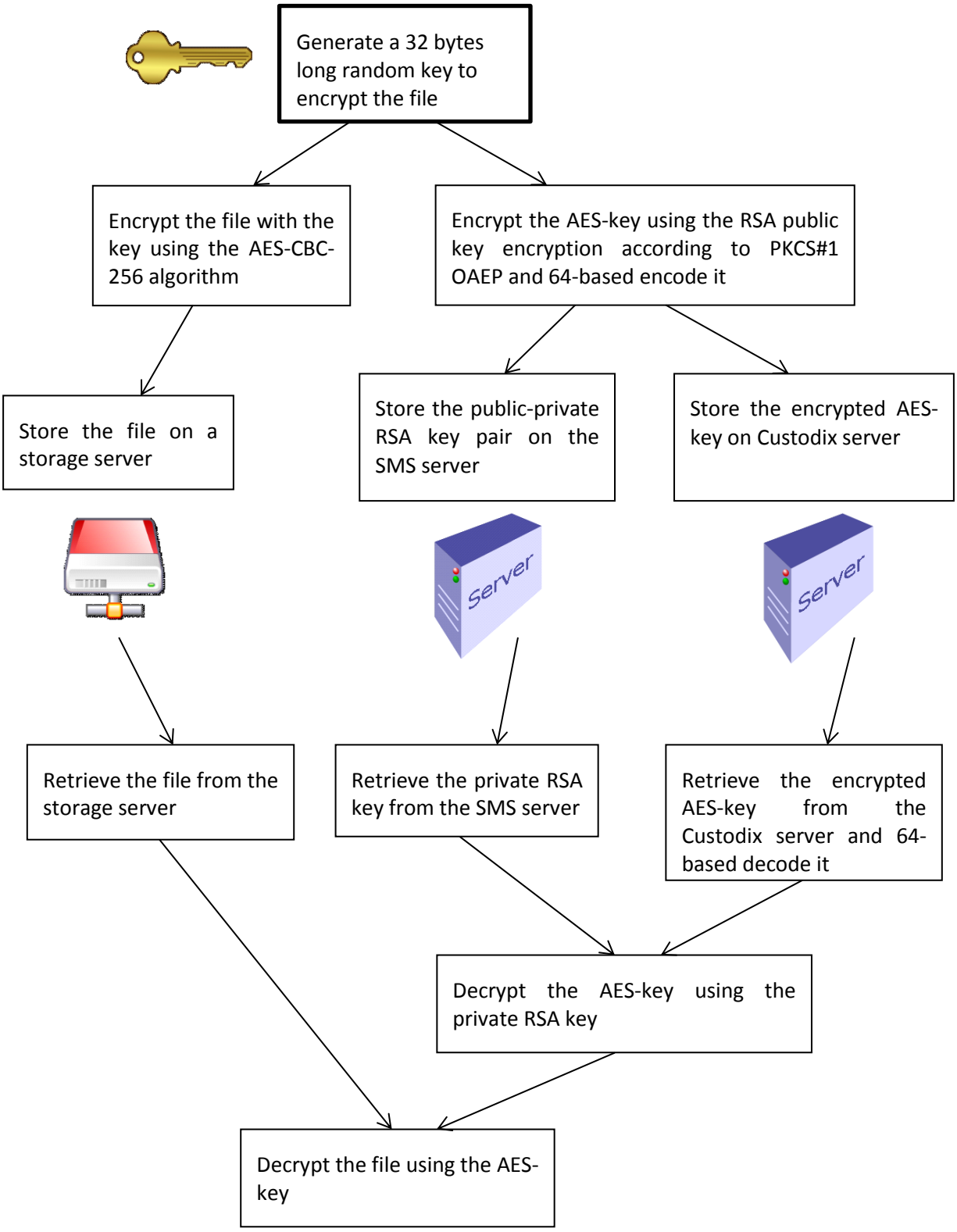


Figure 2 – Schematic representation on the CHIC encryption mechanism

4 Conclusion

In this technical report, a description of the encryption services implemented in the CHIC VPH-HF infrastructure is provided. We decided to use a symmetric-key algorithm to encrypt the data files and a RSA public asymmetric-key algorithm to encrypt the key. For security reasons, the key and the encrypted files are stored in different servers, as well as the RSA private-public key pair. Custodix has provided a web-service to store and retrieve the RSA-encrypted key.

The encryption services are already integrated and available as part of the Storage Management Service of the VPH-HF release described in D7.2 [1].

5 References

[1] D7.2 – First Release Hypermodelling Framework

[2] http://en.wikipedia.org/wiki/Encryption

[3] https://www.physiomespace.com/

[4] http://en.wikipedia.org/wiki/Public-key_cryptography

[5] https://www.dlitz.net/software/pycrypto/

[6] http://www.ietf.org/rfc/rfc3447.txt

[7] D5.2 - Security guidelines and initial version of security tools

Appendix 1 – Abbreviations and acronyms

AES-CBC	Advanced Encryption Standard-Cipher Block Chaining
REST	REpresentation State Transfer
SAML	Security Assertion Markup Language
RSA	Initials of the surnames Ron Rivest, Adi Shamir, and Leonard Adleman; it is a public-key cryptography algorithm
RFC	Request for Comments
SMS	Storage Management Service
OAEP	Optimal Asymmetric Encryption Padding
VPH-HF	Virtual Physiological Human - Hypermodelling Framework
IdM	Identity Management component