Engineering Secure and Private Systems Using Computer Supported Security Patterns

ANTONIO MAÑA, University of Malaga
JOSE FRAN. RUIZ, University of Malaga
MARCOS ARJONA, University of Malaga

The design and development of secure systems is a very complex task for non-expert security users because the integration of requirements, properties, constraints and functionalities of Security and Privacy (S&P) artifacts in systems requires a high level of expertise and knowledge. This task is already a challenge for an expert but for a non-expert is very difficult. One way of minimizing the complexity is by using security artefacts such as security patterns. Although traditional Security Patterns (SPs) offer many functionalities that help in this task, they have limitations that do not facilitate its use by non-experts. To overcome those limitations, we proposed a new type of SP called COmputer-Supported Security Pattern (COSSP) that aims to cover all the gaps of traditional SPs by extending them with new contents, making them machine-processable, adding engineering oriented extensions and providing trust-management capabilities. In order to demonstrate how COSSPs facilitate and enhance the creation of secure systems this paper illustrates how it is applied and integrated into a system model during the engineering process of a given SUD. The integration covers both the design and the development phases. Therefore, our paper will present the benefits of our approach in creating a secure system and how it improves the traditional security patterns functionality in the different stages of an engineering process.

Categories and Subject Descriptors: **D.2.11 [Software Engineering]**: Software Architectures—*Patterns*; **D.3.3 [Language Constructs and Features]**: *Patterns*; **K.6 [Management of Computing and Information Systems]**: Security and Protection

General Terms: Security Patterns, Security modeling and engineering, Pattern-based system engineering

Additional Key Words and Phrases: Computer-processable knowledge representation

1. INTRODUCTION

The design and development of secure systems is a very complex task for non-security experts because the integration of Security and Privacy (S&P) artifacts requires a high level of expertise and knowledge in order to follow the principles determining how to use security properties, constraints and functionalities to meet security requirements. This task is already a challenge for an expert but for a non-expert is almost impossible. One way of minimizing the complexity is to use security artifacts that provide all the necessary information and functionality of the security properties to be used in a system with descriptions and back-up support for applying the corresponding mechanisms that provide their solution.

Although traditional Security Patterns (SPs) can offer some of these characteristics, they have limitations that do not facilitate its use by non-experts. Among these limitations, the most important ones are lack of support for computer-assisted design, cross referencing, assisted integration of the elements belonging to the pattern in the System Under Design (SUD), consistency checks, trust mechanisms, verify interaction conflicts in multi-pattern solutions, etc. To overcome those limitations, we proposed a new type of SP called Computer-Supported Security Pattern (COSSP) [1] that aims to cover all the gaps of traditional SPs by extending it with new contents and features, making them machine-processable (for creation, management, searching, integration, etc.), adding engineering oriented extensions and providing trust-management capabilities. We encourage readers to check the COSSP definition presented in [1] as we will use elements from there that will not be fully explained in this paper.

COSSPs architecture allows a holistic approach to computer and user friendly engineering processes. It was first presented in PLoP'13 and currently is being improved and modified in order to cover all the requirements and functionalities provided by users and taking into account all their feedback comments about usability. The

This work is supported by the CUMULUS and PARIS EU projects.
Author's addresses:
Antonio Maña, University of Malaga. Boulevar Louis Pasteur, 35, 29071 Málaga, Spain; email: amg@lcc.uma.es.
Jose Fran. Ruiz, University of Malaga. Boulevar Louis Pasteur, 35, 29071 Málaga, Spain; email: joseruiz@lcc.uma.es.
Marcos Arjona, University of Malaga. Boulevar Louis Pasteur, 35, 29071 Málaga, Spain; email: marcos@lcc.uma.es.

COSSPs are currently being tested to manage and solve security requirements, which arise between local systems and cloud platforms, providing a combination of local assurance measures with certification-based solutions in the EU CUMULUS project [2].

COSSPs aim to help engineers in the creation of security patterns and end-users on its selection and deployment in their systems. On one hand, a COSSP provides a useful structure for defining security properties which cover assets, the SUD elements with which it interacts, the solution mechanisms (including also models that will be used in the system model under development), post-process validation activities, etc. On the other hand, regarding end-users, COSSPs are enabled for a very intuitive and easy integration in system models. As they are computer-oriented the search is very intuitive (e.g. keywords, security property to be fulfilled, assets to be protected, etc.) and they can be applied with the models and other solution artefacts they include. This implies that, although end-users have no knowledge or experience in the security field, they can work with the automated elements of the COSSP and obtain its benefits. Therefore, in order to keep the paper focused in the use of the COSSPs in a system model we describe briefly their main characteristics:

- Engineering-oriented: they enable an easy integration in engineering activities and processes, facilitating the task of adopting security patterns. COSSPs are also designed with the capability to work and coexist with current processes and methodologies, allowing engineers to use them without the need of changing their usual development procedure or platform.
- Machine-processable: COSSPs are designed to be treated by computer-assisted means for their creation, classification and cataloging for proper repository storage or retrieval. Besides, they can be used and integrated in modeling and development environments.
- Trusted: COSSPs provide trust management and signature mechanisms that facilitate their use by system engineers. This allows the creation of trusted communities that improve greatly the use of security patterns assuring the chain of trust between pattern designers and system developers.
- Detailed and precise: COSSPs allow security experts to capture and represent a rich set of characteristics and details with enough rigor and precision to suit the needs of different users and scenarios but without sacrificing their generality, detaching the security knowledge from very specific techniques, which is necessary in order to ensure their reusability. This is possible thanks to how the information of the security solutions is specified in the COSSP and the tool used for applying them.
- Open and decentralized: COSSPs allow security experts to publish their patterns in an open way allowing other collaborators to analyze, improve them or even using them for creating a new branch with different characteristics. Their computer-oriented nature facilitates this task and provides means for communicating changes and updates to users.

COSSPs provide detailed information about the implementation of the security solutions they represent and the security properties they provide. That part comes very handy in the final stages of the engineering process, as the impact of these solutions in the system is already taken care of (e.g. extra elements they need such as keystores, adaptors, interfaces to connect different elements, etc.). Finally, COSSPs are developed by security and privacy experts with the intention to be used by experts and non-expert users. This means that the COSSPs supporting tools provide many functionalities and information that help users in their management. Also, the COSSP-related tools allow engineers to create the patterns and to use them in their system models in order to fulfill their security requirements. The COSSPs are applied semi-automatically to the system model and provide information about the relations of the security elements with the system, external elements, etc.

COSSPs are stored in a public repository that acts as knowledge base for security solutions. Engineers can access and search for the most useful patterns that better fits their necessities and requirements.

This paper illustrates how a COSSP is applied and integrated into a system model of a given SUD following an engineering process. The integration covers both the design and the development phases. In the design phase, the different UML diagrams represents the artefacts and functionality of the security property to be applied to specific assets, and how it should be deployed in the user's system model. For example, once the class diagram is automatically integrated in the SUD, users visualize the new elements of the system, the requirements they have (e.g. other COSSP or additional certification mechanisms), etc. That way the user can start, in this phase, taking actions in order to cover all the new necessities of the system. This allows systems to be more resilient and complete in design time, being this characteristic a big improvement compared with actual approaches where security is integrated in a latter stage and thus modifying all the initial structure of the system model (and making its design phase usually a waste of time). In the development phase, after all the security requirements of the system have been fulfilled with the S&P properties of the COSSP, users can use the solution mechanisms specified in the implementation part of the pattern and take advantage of its supporting

functionalities: using the OCL rules to verify the correct integration of the pattern in the system, tests to verify if the solution is correctly implemented, etc.
In order to better demonstrate how to use COSSPs in an engineering process of a system we provide an use case example where we describe some of its S&P requirements and how we apply COSSPs in order to fulfill them, showing the extended models and information for its integration. Following we discuss the results presenting its benefits and limitations.

2. APPPLYING COSSPS FOR ENGINEERING SECURE AND PRIVATE SYSTEMS

This section presents how we apply COSSPs to a system in order to incorporate all the security enhancements required for their S&P necessities. We are going to describe here the use case together with the security requirements we need to fulfill, the security patterns we are going to use (in their current standard form and their COSSPs version), the integration of the patterns to the system and finally the benefits and limitations of the approach.

2.1 Example Scenario

The design of systems to communicate official government information to citizens is a very complicate task from the point of view of security and privacy. Such systems must fulfill very strict policies that assure the integrity and confidentiality of the messages, provide secure two-way authentication so that both citizens and the government can have guarantees about the sources of the messages they receive, it must provide non-repudiation, etc. All these security properties use external elements of the system that interact in several ways with it (e.g. keystores, ports, secure channels, etc.). Therefore the complete system must be provided by means of trusted security artefacts. Trust benefits from informal aspects like being developed by knowledgeable and experienced developers, adopting solutions proven in other similar problems, being successfully tested in high-risk environments, etc. Figure 1 presents an abstract description of the e-Government application and the security properties it must provide (some of them).

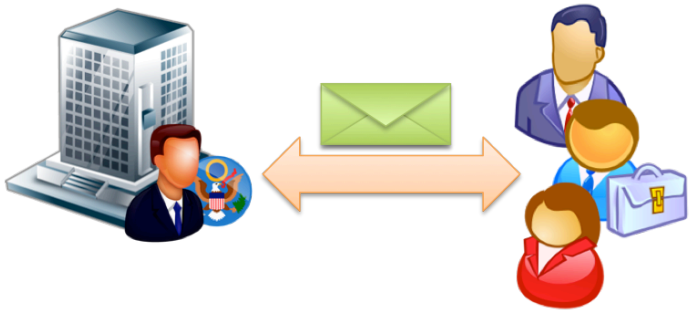

Figure 1. e-Government Trusted Messaging

That way, after performing a security requirements analysis we identified, among others, the following security requirements of the system:

- Confidentiality of the content of the messages
- Privacy of the existence messages
- Authentication of the users
- Authentication of the government agencies and officials
- Non repudiation of the sending and reception of messages

In order to simplify the engineering of the example using our approach, in this paper we are going to focus on two security requirements: confidentiality of the content of the message and authentication of the sender.
We are going to use traditional security patterns and their COSSPs equivalent in order to demonstrate how we design the system and provide the security properties that fulfill the abovementioned requirements. In this case we are going to use an encryption pattern for achieving the confidentiality of the message and a digital signature pattern for the authentication of the sender. The first one will give us the tools for encrypting and decrypting the message sent from the government to the citizen and the last one will provide a digital signature functionality for the government so citizens can verify the authenticity of the message.

2.2 Patterns

The patterns we are going to use as basis for the engineering of our approach are “Symmetric Encryption and XML Encryption Patterns” [3] and “Digital Signature with Hashing and XML Signature Patterns” [4].

The encryption pattern provides information about how to create a secure communication functionality in a system using cryptography. The external element that the system needs to import is a keystore (where the encryption/decryption keys are stored). The pattern includes information about how the solution works and a class diagram showing the functionality and interaction of the different elements. Figure 2 shows the class diagram of the pattern and its relations.

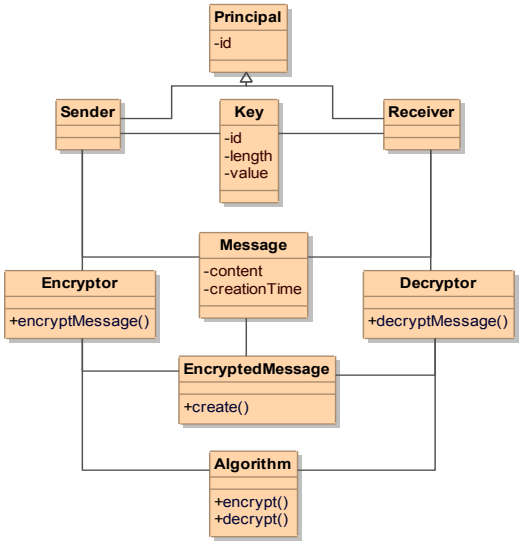

Figure 2. Encryption pattern class diagram

As we can notice the class diagram describes the elements for the solution but not how it interacts with a system or the elements of the system that are related to the ones of the solution. This is intended, as we only want to show the relations between the elements of the solution and their descriptions. Another field it includes is the implementation, where the pattern describes different solutions that can be used for providing confidentiality to a messaging functionality. Unfortunately the solution is human-oriented description and doesn't provide a solution equivalent and tailored to the previous information it provides. This is a problem when the development phase comes, as developers will not know how to integrate the solution in the system and how to know if it fulfills the needs or even if further consequences and dependences are also attached to the SUD once the pattern has been fully deployed.

The digital signature pattern provides information about how to digitally sign a message so the receiver can verify that the sender is who it claims to be. This is a very important element for our scenario because the users need to trust in the messages they receive. The pattern we use as example describes the problem and includes a class diagram showing how the solution works. Figure 3 shows this diagram.

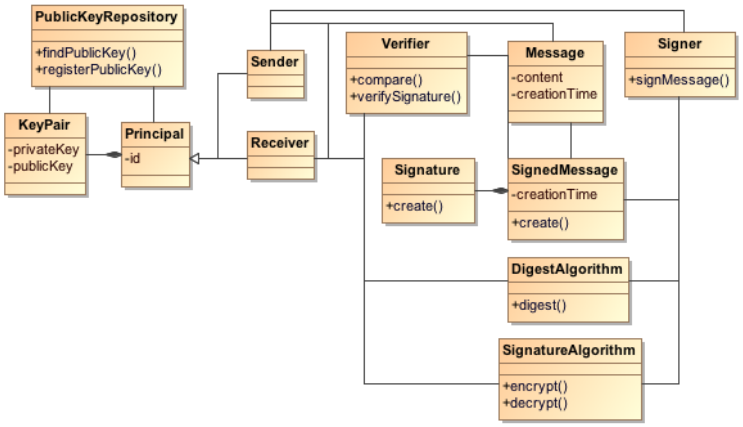

Figure 3. Digital signature class diagram

Same as before, although the diagram describes the elements for its functionality and relations it doesn't include the relation with the elements of the system or the external elements that must be included. In this case it should inform that the system where this pattern is applied requires a keystore that has to be accessed by

both the sender and receiver (for obtaining the private and public key). The solution and implementation are done at an abstract level too. While they help to understand how the solution works the developer will have problems when implementing the system (how does the pattern interacts with the SUD?, how is the solution functionality provided by the structure described in the pattern?, is any reliable implementation suggested?, etc.).

One problem we foresee when working with these patterns in our use case example is that both of them use a keystore and, when applied to a system, engineers will not know that both of them can use the same. This has an impact at the design and development level. At design time, we lack a way to know in our system model that both keystores can be used as just one, as the two patterns allow this. At development level this characteristic can facilitate the implementation and generate better and optimized results.

We have created COSSPs versions of these same patterns in order to show how COSSPs enhance the classical patterns. Due to the length of the paper and because most of its information is computer-oriented we do not include here all the information of the COSSP. Regarding the class diagram of the solution the COSSPs provide the SUD elements and assets that interact with the functionalities or artifacts of the solution provided by the pattern. The COSSPs tool also allows to create the patterns in a way that they can be compared and searched for specific fields.

Figure 4 presents an excerpt of the class diagram of the encryption pattern described before in COSSPs format. The "Key" element described in the diagram implies the use of a "Keystore" asset (showed in the diagram in the relation between "Key" and "KeyStore"). This characteristic is created using the information of the classical pattern we are using as basis. If it doesn't exist in the system model it will be created by the tool used for applying the pattern. The "Sender" and "Receiver" elements are marked as interfaces due to their final role in the SUD, responsible to operate and coordinate all the encryption/decryption mechanisms in the secured software. That way the engineer will know how to integrate these elements to the ones of the system model and will understand the purpose of all the components of the pattern.

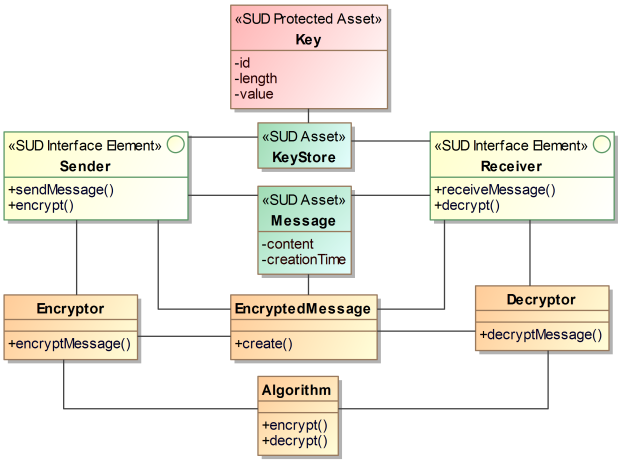

Figure 4. COSSP encryption pattern

Therefore, engineers are able to easily integrate the solution of the pattern in their system model. Besides, as COSSPs are computer-oriented, when the engineer applies a pattern to the SUD the class diagram is automatically deployed, adding connections with the elements of the system. This also brings another benefit to the engineer and solves many design issues such as the problem we described previously: the use of the same keystore by the two solutions. Thanks to the automatic generation and integration of the functional behavior in the design phase, when the engineer integrates the first pattern using a tool it would include the additional keystore in the system. Following, when the expert integrates the second pattern (digital signature) in order to fulfill the "Authentication of the sender" requirement, the tool will search for an already existent keystore in the SUD and, when it finds it, it will reuse it for the digital signature solution.

The application of classical Security Patterns in the same way would be very different. First of all, as they are user-oriented they are mainly documents and diagrams that are used only as a guide of how to apply the pattern to the system. It is responsibility of the user to apply the different elements of the class diagram to the SUD, modify the necessary elements, define the relation between the elements, etc. Also, the user will need to analyze and include the new elements the system model needs and check if new security threats appear (and how to secure this new security threats). Also, in a case where two patterns use the same element the user has

to research if they have the same functionality and objectives. Therefore, the application of a classical SP is very manual and depends a lot of the expertise of the user, while our approach is supported by tools.
The COSSPs versions of these patterns also extend the implementation part by providing a series of rules (in OCL) that can be used to verify the correct integration of the pattern in the system and tests that can be used to verify that the pattern is correctly implemented. Besides, it can include references to existing suggested implementations, providing real used mechanisms capable to address the security requirement.

2.3 Supported integration of COSSPs in the SUD

The e-government use case example shows a clear separation of responsibilities between the client and the server. This scenario requires more than one unique profile to express the overall architecture. It needs to reflect the complete functionality involved in the SUD by modeling components both for the local software and also for the corresponding elements in the remote servers to interact with. Figure 5 shows an excerpt of the scenario with these two different groups of components.

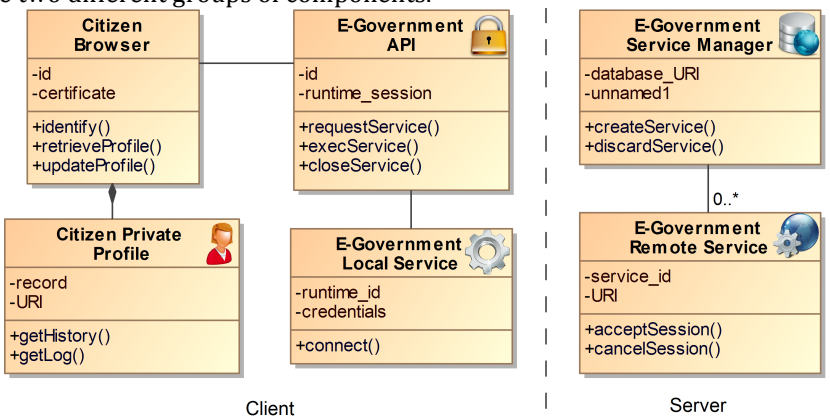

Figure 5. e-Government System Model

The usual approach for e-government communications forces users to install proprietary software either as an external application or as an add-on in the users' Internet browser. Once the authenticity of the software has been proven and the extension has been added to the parent application, users are enabled to interact with the government administration in order to use all the offered services. A local service dialog with an e-government remote service provides a secure and protected communication thanks to the use of credentials and strong certification mechanisms. Therefore, our example focus in the application of the encryption COSSP in the system model in order to guarantee a secure communication between these two services.
The users use a tool for applying the Encryption COSSP to the system model for fulfilling the "confidentiality of the message" requirement. According to the Encryption COSSP specification the tool asks users the main elements where the pattern interacts such as the interface and the assets in order to identify them and apply the required elements and connections to them. In our case, the users identify "Sender" as E-Government Remote Service, and "Receiver" as E-Government Local Service. They also choose to create all the necessary elements requested by the pattern because the SUD does not contain any keystore, encryptor, decryptor or any message entity. Once the automatic process has finished, the enhanced system model is shown in Figure 5. We can notice all the changes and modifications introduced to provide the Encryption functionality.

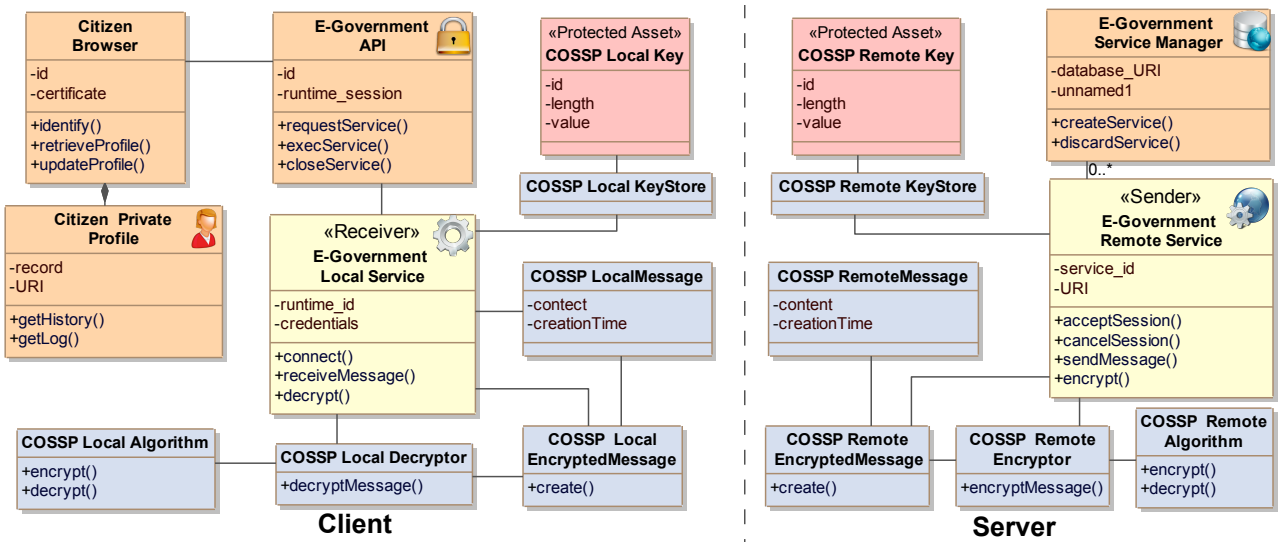

Figure 6. Encryption pattern applied to System Model

The deployment of the Encryption COSSP requires not only adding new elements but also to integrate them in the system model and include the necessary mechanisms in the current components of the system. In our case, the patterns describe two main interfaces, "Sender" and "Receiver" with their particular.

The pattern applied in the system model works as secure sending functionality from the server to the client. Therefore, although the pattern provides methods for encrypting and decrypting the client will only decrypt the messages received from the server, which will use the pattern to encrypt them. Therefore, both parts of the system (client and server) have access to the COSSP Local Algorithm but each of them uses it in a different way (only specific functionalities).

Therefore, it is mandatory to integrate these operation in the selected elements: "E-Government Local Service" is expanded with "sendMessage()" and "encrypt()" operations and "E-Government Remote Service" with "receiveMessage()" and "decrypt()". The additional elements added to the system models are labeled with the "COSSP" root to identify them. They appear replicated in both sides, local and remote, because they cannot be shared and it is necessary to reproduce the expected functionality.

Additionally, we can notice that the key elements, in red, have the "Protected Asset" mark. Due to the machine-processed sense of the COSSP, the designer included the suggestion to use an additional security mechanism in order to protect these assets. In our example the expert included the usage of an additional pattern ("Secure Storage COSSP") in the specification as a derived reference. Therefore, system engineers could use the COSSP tool to retrieve this pattern from the repositories and deploy it in the system model as well, if they are interested and they consider the importance of this new security requirement.

Finally, COSSPs also integrate in the system model a series of validators in form of OCL rules that help to verify the proper integration of the pattern. Any potential conflict will be discovered by these constraints and notified to the users in the modelling framework. Besides, these rules also target other potential issues like unauthorized modifications of the pattern structure or even help to avoid potential overlapping problems when additional new patterns are deployed to the same system model.

3. DISCUSSION

The previous section describes how COSSP are deployed in a SUD in order to obtain enhanced system models with the pattern components integrated in the system architecture. In contrast with the standard SPs, this procedure is fully automatized and for that reason, COSSP requires additional extensions and improved mechanisms. Non-expert security engineers are able to apply and take advantage of the benefits COSSPs brought by using tools that process and use their information. Also, by having a system model with all the necessary artifacts at design time allows for a better work in the development time. Users always receive help from the modelling framework and all the SUD changes are done automatically and validated at design time. Moreover, all the involved security knowledge belonging to the used COSSP is available in the modelling tool allowing users to have the information, advantages and drawbacks of the solutions in a very friendly way.

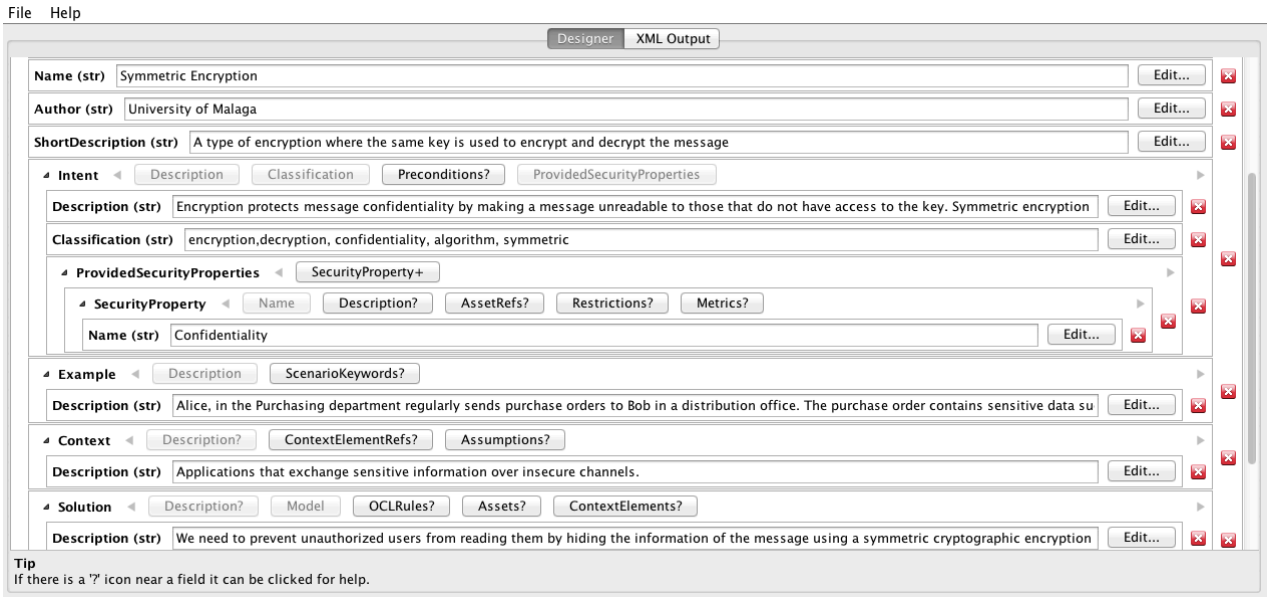

Figure 7. COSSP tool editor view

The COSSP tool aims to support the design and implementation phases of the engineering process. Figures 6 and 7 show some of its functionality in action. Figure 7 shows the COSSP editor. This functionality allows engineers to create the pattern information in a computer-oriented way. Once the pattern is completed it is stored in a repository that can be seen in Figure 8. The repository has a search functionality that filters the patterns and shows their information.

Regarding the support of the phases of the engineering process on one hand, at design time, the tool allows engineers to apply the patterns as we showed in the previous section. The tool is integrated in a natural way with UML design tools such as MagicDraw, taking advantage of its mechanisms and characteristics. That way, users can search and import COSSPs to the framework and apply them to their system models. We are also developing an Eclipse-based tool so there exists more options for developers. On the other hand, at development time, COSSPs provide links and information about different solutions that provide the security property and functionality of the pattern. This means that users can access this information directly and use it in their system. Last, COSSPs have information about testing that can be used in a system where they are applied in order to check the resilience of the system against the attacks that can target it. The testing information is practical artifacts such as JUnit code, UML sequence diagrams representing failure cases, etc. They are created by the COSSP engineer, who has the knowledge and expertise for creating them, as she knows what are the main threats for a specific security property and how they must be tested. Therefore, COSSPs are designed and created in a way that they and the supporting tool can be used easily with other existent engineering processes, facilitating its use and functionality.

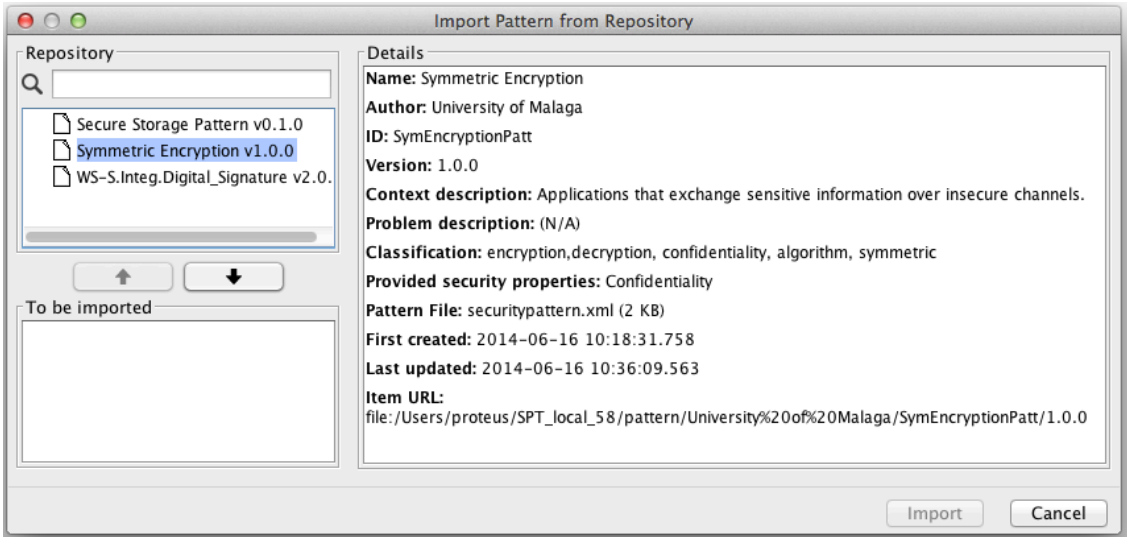

Figure 8. COSSP tool import

SPs establish a different approach. As all the mechanisms and modelling activities require the interaction and knowledge of the engineer it demands a high level of expertise. This increases the process complexity by imposing system engineers to have a minimum level of knowledge in order to deal with all the functional and non-functional details of the SPs and also prevent collateral results such as the overlapping of patterns, replication of components with the same function, avoid derived implementation collisions, etc.

The vision of a SP usually offers a static vision of a well-known and widely documented solution but in current system engineer activities security has become a very evolving matter with multiple considerations to be aware of. In these risky environments, where S&P requirements are of vital importance, security artifacts must provide updated solutions, integrate trusted assurance mechanisms and deploy post processing techniques like monitoring or evidence based tests for verifying the proper execution of the software. Besides, these activities must be reactive to constant changes in order to deal with emerging threats and undiscovered threats.

In this sense COSSP are adequate to address those necessities. These security artifacts have been structured to supply users evolving security mechanisms based on reliable SPs and tailored to specific system conditions. The creation and managing of the patterns are also enhanced with the COSSP approach. All the patterns are created using a pattern template that is automatically indexed, allowing for an easy evolution to new versions with different elements. The search and use is also automatized, using online repositories that have all the patterns indexed in their database. That way, users can use tools for searching, downloading and applying a pattern automatically. As we have shown in the use case example the patterns provide models that describe their structure and use and have information about the SUD elements they are related to (or even external elements). The solutions included in the COSSPs follow these diagrams. Therefore, when an engineer have finished designing her system model the next step, development, is easier because the solutions and artifacts follow exactly what is defined in the design phase.

REFERENCES

[1] Mana, A. Fernandez, E.B., Ruiz, J.F. and Rudolph, C. Towards Computer-oriented Security Patterns *The 20th International Conference On Pattern Languages Of Programs PLoP'13*. 2013

[2] Arjona, M., Ruiz, J.F and Mana, A, Security Patterns for Local Assurance in Cloud Applications. *International Workshop on Engineering Cyber Security and Resilience ECSaR'14 in The Third ASE International Conference on Cyber Security 2014.*

[3] Keiko H. and Fernandez, E.B. Symmetric Encryption and XML Encryption Patterns. Dept. of Computer Science and Engineering, Florida Atlantic University Boca Raton, Fl, 33431, USA,

[4] Hashizume, K., Fernandez, E.B. and Huang, S.. Digital Signature with Hashing and XML Signature Patterns. Dept. of Computer Science and Engineering, Florida Atlantic University