A Security Pattern for Untraceable Secret Handshakes

Angel Cuevas
Department of Telematic Engineering
University Carlos III of Madrid
acrumin@it.uc3m.es

Paul El Khoury
LIRIS University of Claude Bernard Lyon I
SAP Research Sophia-Antipolis
paul.el.khoury@sap.com

Laurent Gomez, Annett Laube
SAP Research Sophia-Antipolis
laurent.gomez@sap.com, annett.laube@sap.com

Alessandro Sorniotti
Institut Eurecom Sophia Antipolis
alessandro.sorniotti@eurecom.fr

Abstract

A security pattern describes a particular recurring security problem that arises in specific contexts and presents a well-proven generic solution for it. This paper describes an Untraceable Secret Handshake, a protocol that allows two users to mutually verify another's properties without revealing their identity. The complex security solution is split into smaller parts which are described in an abstract way. The identified security problems and their solutions are captured as SERENITY security patterns. The structured description together with motivating scenarios makes the security solution better understandable for non-security experts and helps to disseminate the security knowledge to application developers.

1. Introduction

Today's pioneer organizations recognize that performance accelerates when information security is driven into the very framework of a business. Security experts developed and standardized sufficient security solutions to satisfy security requirements for specific contexts. Most of these standards provide comprehensive methodologies for specifying, implementing and evaluating security of IT products. Rarely non-specialists are capable of correctly interpreting such standards. Several standardized methodologies were developed to support the development of secure systems. Most of these standards provide comprehensive methodologies for specifying, implementing and evaluating security of IT products. Unfortunately, security experts are the interpreters of such standards. The description of these standards in natural language limits their ability in passing knowledge to novice security users. The fundamental added value of adopting the security patterns approach is providing security for non-security experts [23][25].
Application developers transform clients' requirement specifications into business applications. Business solutions and security solutions are often designed and developed at different coordinates of space and time. Considering the lack of expertise application developers are in general incapable of being compliant to security regulations protecting the clients' business.

Security patterns capture security expertise abstract and concrete at the same time. This approach fits well as candidate link between security experts and application developers to encompass business applications with a security shield.

This paper focuses on capturing the security solution for Secret Handshakes described in [2] and [24] as security patterns, following the SERENITY methodology [16][25]. Parties cooperating in hostile networked environments often need to establish an initial trust. Trust establishment can be very delicate when it involves the exchange of sensitive information, such as affiliation to a secret society or to an intelligence agency.

Secret Handshakes are first introduced in 2003 by Balfanz et al. [2] as mechanisms designed to prove group membership and share a secret key between two fellow group members. The purpose of these protocols is – as pointed out in [26] – to model in a cryptographic protocol the folklore of real handshakes between members of exclusive societies, or guilds.

Even though there are several ways for implementing the secret handshake protocol, our purpose is to capture the properties and functionalities that are common to all implementations. Therefore, any particular solution could be derived from the defined patterns. Furthermore, capturing expertise as a pattern makes security solutions for a given problem more general. It is easier for non-security experts to find a suitable solution for a particular problem by search-

ing into the patterns' library through properties and features.

The paper is organized as follows. In Section 2, we overview Security Patterns. Section 3 proposes two scenarios to illustrate the use of secret handshakes in real-world applications. We describe the proposed security solution detailed in Section 4 and define an abstract model used to capture this security solution as a combination of patterns. Next, Section 5 describes the security patterns and integration scheme of the proposed solution in more detail. Related works are discussed in Section 6 and finally in Section 7 we conclude and explain future work.

2 Security Patterns Overview

To accomplish the security patterns' 'mission', a list of objectives summarized in four fundamental steps [23] has to be achieved. First, most of the novice security users should understand how experts approach key security problems. Second, security experts should be able to identify, name, discuss and teach both problems and solutions efficiently. Third, problems should be solved in a structured way. Fourth, dependencies and side-effects should be identified and considered appropriately. The connotation of these objectives emerged as appealing for research studies.

The usual natural language description for security patterns opens room for different interpretation of solutions provided and problems described by these patterns. Hence, none of the previously four objectives of the patterns' mission can be achieved.

First known contribution to security patterns, is the work from J. Yoder and J. Barcalow proposing to adapt the object-oriented solutions to recurring problems of information security [28]. Seven patterns were presented to be used when dealing with application security. A natural evolution of this work is the proposal presented by Romanosky in [19]. It takes into consideration new questions that arise when securing a networked application.

Following this particular path, Schumacher et al. [22] presented a set of security patterns for the development process. Fernandez and Pan [12] describe patterns for the most common security models such as Authorization, Role-Based Access Control and Multilevel Security. Recently in [13] the same authors highlighted the need to develop additional security patterns for database systems in order to integrate it into secure software development methodology. These security patterns had a shy adoptions in the security field. Indeed their description in natural language limits their applicability and forbid any reasoning mechanism.

The SERENITY EU project through a list of narrow yet complex studies [25][16][8][21] tackles the security patterns objectives.

The SERENITY partners presented in [25] the SERENITY model of secure and dependable applications. Moreover, using security patterns they showed how it addresses, along with the tools provided, the challenge of developing, integrating and dynamically maintaining security mechanisms in open, dynamic, distributed and heterogeneous computing systems.

One of the essential proposals from SERENITY is to provide novice users the SERENITY Security & Dependability pattern package. This package comprises the *expert-proofed* security solutions and *tested* plug-and-play deployable implementations.

The research interest in security patterns focuses in particular on capturing solutions for recurring security problems that arise in specific contexts. The granularity of security problems analyzed and captured in a pattern, can be quite different. Usually a complex security solution is not captured in a single pattern. Solutions consisting of several patterns cover better the generality aspect of the abstract solution.

To have an intuitive description for the solution proposed in this paper, we adopt the SERENITY approach using three artefacts. The description of these artefacts enable selection, adaptation, usage and monitoring at runtime by automated means. The hierarchy is composed by three artefacts, Security Classes, Security Patterns and Security Implementations. Although this paper emphasized the use of Security Pattern artefact, [20][3][9][5] present an intuitive and extensive description of all of them.

In SERENITY, *security patterns* are detailed descriptions of abstract security solutions that contain all the information necessary for the selection, instantiation and adaptation performed on them. Such descriptions provide a precise foundation for the informed use of the solution and enhance the trust in the model.

An *integration scheme (IS)* is an additional artefact defining the combination of security patterns. Since complex solutions rely on the use of several patterns, they have to be defined as integration schemes. In the IS, the relations among the patterns are established in order to describe a complex security solution.

This paper relies on the SERENITY representation of security patterns [21][25] to transfer the first three objectives of security patterns for the Untraceable Secret Handshakes to non-security experts. The most important parts of a security pattern description are in the following:

Problem/requirements and context: The problem is the vulnerable part in an asset that can also be described as requirements which need to be solved. The context defines the recurring situation where the problem/requirement can occur.

Solution: The solution is defined as a mechanism that is used to resolve the corresponding requirement/problem. It defines the sequential flow of operations in solving the security problem.

Pre-Conditions: They indicate assumptions and restrictions related to the deployment of the pattern. Before applying a pattern, users or applications in some cases should check the satisfyability of these pre-conditions. Obviously, pre-conditions are elements used during the selection of suitable patterns for a particular problem.
Properties: They describe which security elements the pattern is providing. This is the basic element used to discriminate whether a pattern is useful for a security problem or not.
Features: They are additional characteristics to the patterns' properties. They are additional criteria in selecting the suitable patterns.
Consequences: They are the effects of the compromise resulting from the application of the pattern's solution. In particular cases, using security patterns implies an increase in cost (economic, more complex mechanisms, etc.).

3 Scenarios

In this section, we want to show how untraceable secret handshakes are used in real-world applications. Our first example comes is an use case from the EU Project R4eGov [11] for **Mutual Legal Assistance** in international crimes. Several EU justice forces cooperate in order to solve cross-boundary criminal cases. EU regulations define official processes that must imperatively be followed by operating officers: in particular, these processes mandate which institutions must cooperate upon each particular case. During such collaboration, for instance, a member of France's *Ministère de la Défense* must cooperate with a member of the *Bundesnachrichtendienst*, Germany's intelligence service, to investigate on an alleged internal scandal. The two officers may need to meet secretly, and authenticate themselves on-the-fly. Both are definitely reluctant to disclose their affiliation and purpose to anybody but the intended recipient.

A scenario from another business domain is the **Incompatible chemicals in proximity** use case from the CoBIs project [7]. Let us assume that drums are stored in a warehouse; each drum contains a liquid chemical and is equipped with a wireless sensor that is able to perform a secret handshake with other sensors in proximity. Drums can contain some reactive chemicals: the proximity of these drums must be considered dangerous. The goal is to generate safety-critical alerts based on an untraceable secret handshake. The drums get property credentials related to their containing chemicals and a list of references to match the reactive liquids. After a successful matching, an alert is generated and sent to the storage manager. The security features of the secret handshakes described in [24] allow to exchange information of the containing chemicals without revealing them on a wireless channel. Drums with dangerous contents can not be identified or traced.

4 Solution Description

A *Secret Handshake*, first introduced in [2], is a mechanism devised for two users to simultaneously prove to each other possession of a property, for instance membership to a certain group. The ability to prove and verify is strictly controlled by a certification authority, that issues property credentials and matching values respectively allowing to prove to another user and to verify another user's possession of a property. Users are not able to perform a successful handshake without the appropriate credentials and matching values; in addition protocol exchanges are untraceable and anonymous.

We present a pattern for Untraceable Secret Handshakes with proof of group membership as described in [2] or [24]: users are required to possess credentials and matching values issued by a trusted certification authority in order to be able to prove and to verify possession of a given property. Therefore the certification authority retains the control over who can prove what and who can disclose which credentials. However verification is dynamic, in that it is not restricted to own properties.

A Secret Handshake is performed between two parties, in the following also called users. To carry out a Secret Handshake each user needs a credential and matching values. A **credential** is a certification of the user's property by a trusted entity. The entity responsible for the certification of properties is the **Certification Authority (CA)**. The CA is a trusted entity that after a successful verification of a property grants the user a credential. The process of certifying credentials is captured in the security pattern *Property Certification*. Each user that wants to use the Secret Handshake protocol has to perform this step.

The **matching value** allows a user to verify that the other user has a particular certified property. The user can get one or more matching values from the CA. The CA, according to a set of policies, delivers the matching values to a requesting user after verifying his identity and the context. The process of obtaining the matching values is also described in the security pattern *Property Certification*. The policy with the relationships between property credentials and matching values has to be defined beforehand.

The secret handshake itself is carried out between 2 users and can be repeated infinitely. It consists of 2 parts: the secure match of properties and the proof that both parties possess the same key after the matching.

The secure match is initiated by one of the users, e.g., user A. The user A sends an internal state, e.g., a nonce, to the user B. User B replies with another nonce and his hidden credential, computed from the received state and his property credential. When user A receives the hidden credential from user B, he is able to match it with his matching values (see e.g. [24] for details about the used cryptographic algo-

rithms). To complete the matching protocol, user A computes his hidden credential and sends it to user B. This behaviour is described by the security pattern *Secure Match* and has to be applied to both parties.

After a successful matching both parties, user A and B, own a secret, for instance a key, that can be used to secure the further communication between the two. In order to prove that both parties have the knowledge of the same key the security pattern *Mutual Key Proof of Knowledge (Mutual Key PoK)* can be used. The parties exchange encrypted information to prove their knowledge without disclosing the key directly.[1]

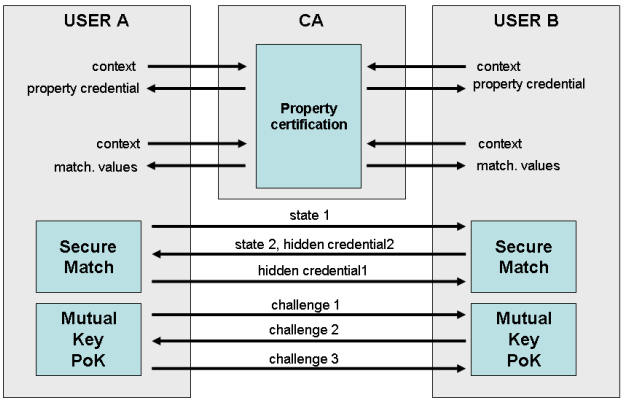

Figure 1. Conceptual Model

Figure 1 depicts the Untraceable Secret Handshake model highlighting the interactions and data flow between the entities and the tasks described by security patterns. The conceptual model helps to understand the dependencies of the security patterns and the interaction of the different actors.

5 Security Patterns and Integration Scheme

Following our abstract model, we identified the following three security patterns: *Property Certification*, *Secure Match* and *Mutual Key Proof of Knowledge*. The integration scheme that describes our security solution entirely is named *Untraceable Secret Matching*. In this section, each of them is defined in detail.

5.1 Property Certification Pattern

Problem/requirements and context: The secure matching is based on the mutual verification of user's properties. In a first step, the possession of a given property has to be verified by a Trusted Third Party (TTP). This TTP certifies the property for an identified user by issuing a credential. In a second step, each user needs matching values. The matching values are given by a TTP according to a policy.
Solution: The pattern defines the following operation:

- **certifyProperty**: The input of this function is the application context of the handshake. The context contains all information needed to decide about the possession of a predefined property, e.g., the user's identity and the process he is envolved in. The possession of the user's property is verified in the given context and if this was successful, a credential representing the property is returned (further named *property credential*).
- **getMatchingValues**: The input of this function is the application context of the matching. According to the policy, the operation returns a set of Matching Values.

Pre-Conditions:

- The entity applying this pattern is a trusted party.
- A list of properties that can be certified and a policy how to match the different properties has been defined.
- Communications channels are secured.

Properties: Certification, Policy Enforcement
Features: None
Consequences: The issued property credentials have been kept secret and stored in a safe place. A revocation list has to be maintained.

5.2 Secure Match Pattern

Problem/requirements and context: A user wants to perform a secret handshake with another user in order to verify that the other party possesses a matching property.
Solution: The pattern defines the following operations. In Figure 2 the protocol between the 2 parties is shown.

- **initiate**: An internal state, e.g., a nonce value, is sent to the other party to initiate the handshake.
- **hideCredential**: A hidden credential is generated from the received state and the property credential of the user and randomized. The result is sent together to the other party.
- **match**: Input of the operation is the received hidden credential from the other party, the owned property credential and the matching values. The operation checks, if the received credentials matches one of the matching values. The result of the match is a key.

[1] Depending on the real setup, also other protocols could be used to prove the knowledge of an identical key. For example, both users could send their keys to a TTP that verifies the keys and returns the verification result.

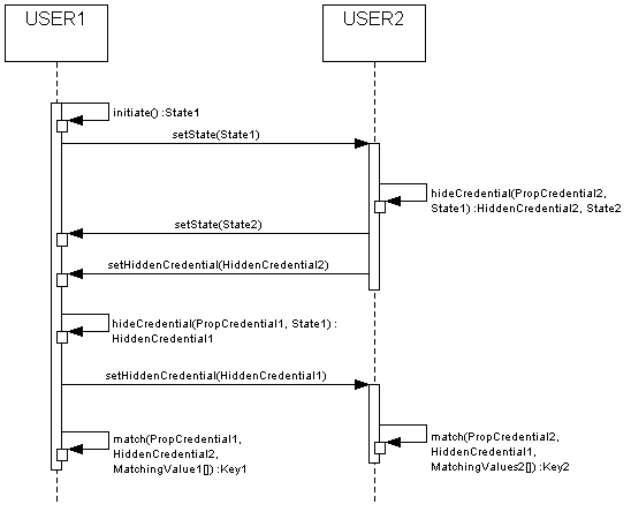

Figure 2. UML diagram - Secure Match

Pre-Conditions:

- The entity applying this pattern possesses a property credential and a non-empty set of matching values.
- Both parties involved in the match must apply the same pattern implementation to ensure interoperability.

Properties: Identification, Property Validation
Features: Untraceability, Key establishment, Fairness
Consequences: None

5.3 Mutual Key PoK Pattern

Problem/requirements and context: A user possesses a secret key. He wants to verify if another user possesses the same key without disclosing his own key.

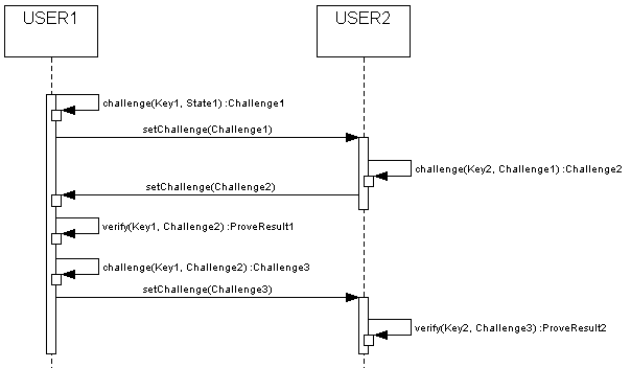

Figure 3. UML diagram - Mutual Key PoK

Solution: The pattern defines the following operations. In Figure 3 the protocol between the 2 parties is shown.

- **challenge**: A challenge is sent to the other party, containing e.g., an internal state (random number) encrypted with the owned key.
- **verify**: The answer to the challenge and the internal state for the current instance of the pattern are processed; this operation returns true in case the two users indeed share the same key.

Pre-Conditions:

- The entity applying this pattern possesses a key.
- A protocol that specifies how to reply to a challenge is agreed on both sides.

Properties: Key verification
Features: Non-disclosure of own key
Consequences: The mutual protocol is unfair since one user knows first if the other party possesses the same key and can therefore exploit his advantage by not replying his side of the challenge.

5.4 Integration Scheme: Untraceable Secret Matching

This integration scheme describes the full solution made of a combination of the defined security patterns. It synchronizes the operations among the patterns in order to provide the desired security solution.

The sequence of the IS operations for the proposed solution are provided in Table 1. We used the keyword $roles$ of the SERENITY security patterns' language [25] in order to separate between the two users of the protocol using the same security pattern.

The IS contains the security property **Mutually Authentication** for the complex security solution. Additional security features of the IS are:

- Resistance to impersonation
- Anonymity
- Untraceability
- Resistance to replay attacks

In addition, this IS describes the list of components (patterns) used for providing the complex solution. All the preconditions defined in the embedded patterns have to be considered before IS can be applied.

6 Related Work

After the introduction of secret handshakes in 2003 by Balfanz et al. [2] many papers have further investigated the subject. New schemes have been introduced, achieving for instance reusable credentials (the possibility to generate multiple protocol exchanges out of a single credential

Table 1. IS Operations

Property Certification Pattern ← CA
Secure Match Pattern ← SMP
Mutual Key PoK Pattern ← MKPP
SMP ← roles: USER1 and USER2
MKPP ← roles: USER1 and USER2
CA.certifyProperty(in:Identity1, in:Context1, out:PropCredential1);
CA.certifyProperty(in:Identity2, in:Context2, out:PropCredential2);
CA.getMatchingValues(in:Identity1, in:Context1, out:MatchingValue1[]);
CA.getMatchingValues(in:Identity2, in:Context2, out:MatchingValue2[]);
USER1.SMP.initiate(out:State1);
USER2.SMP.hideCredential(in:PropCredential2, in:State1, out:HiddenCredential2, out:State2);
USER1.SMP.hideCredential(in:PropCredential1, in:State2, out:HiddenCredential1, out:State3);
USER1.SMP.match(in:PropCredential1, in:HiddenCredential2, in:MatchingValue1[], out:Key1);
USER2.SMP.match(in:PropCredential2, in:HiddenCredential1, in:MatchingValue2[], out:Key2);
USER1.MKPP.challenge(in:Key1, in:State1, out:Challenge1)
USER2.MKPP.challenge(in:Key2, in:Challenge1, out:Challenge2)
USER1.MKPP.verify(in:Key1, in:Challenge2, out:ProveResult1)
USER1.MKPP.challenge(in:Key1, in:Challenge2, out:Challenge3)
USER2.MKPP.verify(in:Key2, in:Challenge3, out:ProveResult2)

with no loss in untraceability) and dynamic matchings (the ability to verify membership for groups different from one's own).

Castelluccia et al. in [6] introduce the concept of CA-Oblivious encryption and show how to build a Secret Handshake scheme from such a primitive. Users are equipped with credentials and matching references (in this particular case embodied by a public key and a trapdoor) that allow them to pass off as a group member and to detect one. In [17], Meadows introduces a scheme that is similar to Secret Handshakes, despite the fact that the security requirements are slightly different – for instance, untraceability is not considered. In [14], Hoepman presents a protocol, based on a modified Diffie-Hellman key exchange [10], to test for shared group membership, allowing users to be a member of multiple groups. In [26], Vergnaud presents a secret handshake scheme based on RSA [18]. In [27], Xu and Yung present the first secret handshake scheme that achieves unlinkability with reusable credentials: previous schemes had to rely upon multiple one-time credentials being issued by the certification authority. However, the presented scheme only offers a weaker anonymity. In [15], Jarecki, Kim and Tsudik introduce the concept of affiliation-hiding authenticated key exchange, very similar to group-membership secret handshakes; the authors study the security of their scheme under an interesting perspective, allowing the attacker to schedule protocol instances in an arbitrary way, thus including MITM attacks and the like. However their scheme is not suitable in our context, since it only allows to verify own group membership and does not consider untraceability of protocol exchanges.

In [1], Ateniese et al. present the first Secret Handshake protocol that allows for matching of properties different from the user's own. Property credentials are issued by a certificate authority.

7 Conclusion and Future Work

In this paper, we described the Untraceable Secret Handshake protocol as SERENITY security pattern (due to the limitation of this paper not all details could be provided). To our knowledge is this one of the first attempts to describe a cryptographic scheme as formal security pattern. While applying the SERENITY methodology to a new area, some weaknesses of the pattern approach became evident. Especially the description of the final solution as integration scheme needs more details than a simple sequence of operations. We suggest to use a standard modeling language, like UML or UMLSec, to describe the complex data flow and interactions in the protocol. This has also the advantage that these languages are well-known by application developers and the diagrams, e.g., sequence diagrams, are easier to understand. Information about operations that can be done in parallel, repeated, omitted or those relying on secure channels, can be easily added.

Another weakness of the pattern approach is the pattern selection process. Can an application developer or architect responsible to implement one of the scenarios from Section 3 identify the right security properties from the use case description or a specification? Security properties like 'Mutual Authentication' with 'Untraceability' have a meaning only for security experts. Here, we see the need to express the security requirements and security properties and features in a way non-security experts understand.

Beside the discovered issues of the SERENITY pattern approach, the abstract description of the cryptographic protocol helps to understand the use of the cryptographic schemes for secret handshakes. These schemes are in general difficult to understand for non-crypto experts and therefore the security pattern approach will help to spread the knowledge in the security community and also among application developers and architects. Following the SERENITY methodology, each security pattern contains one or more possible implementations (*Security Implementations*), which give concrete examples of the solution. In our case, we developed in the context of the WASP project a solution based on web services to mutual authenticate mobile Wireless Sensor Network gateways with different backend applications. A second solutions is based on Ptolemy II [4] and simulates the CoBIs scenario described in Section 3.

In the future, we will further improve the methodology to describe security solutions as patterns. The revocation

scheme for the Untraceable Secret Handshake is a candidate for our next security pattern and will complement the solution described in this paper.

Acknowledgment

This work is partially funded by the European Commission under the Framework 6 IST Projects "Wirelessly Accessible Sensor Populations (WASP)". and "System Engineering for Security and Dependability (SERENITY)".

References

[1] G. Ateniese, M. Blanton, and J. Kirsch. Secret handshakes with dynamic and fuzzy matching. In *Network and Distributed System Security Symposuim*, pages 159–177. The Internet Society, 02 2007. CERIAS TR 2007-24.

[2] D. Balfanz, G. Durfee, N. Shankar, D. K. Smetters, J. Staddon, and H.-C. Wong. Secret handshakes from pairing-based key agreements. In *IEEE Symposium on Security and Privacy*, pages 180–196, 2003.

[3] A. Benameur, P. E. Khoury, M. Seguran, and S. K. Sinha. Serenity in e-business and smart items scenarios. *Security and Dependability for Ambient Intelligence, Series: Advances in Information Security*, Vol. 55, 2009.

[4] C. Brooks, E. Lee, X. Liu, S. Neuendorffer, H. Zheng, and Y. Zhao. Introduction to Ptolemy II. In *UCB/ERL M05/21 Heterogeneous concurrent modeling and design in Java*, volume 1. University of California at Berkeley, 2004.

[5] P. Busnel, P. E. Khoury, S. Giroux, and K. Li. Achieving socio-technical confidentiality using security pattern in smart homes. *The Third International Symposium on Smart Home*, 2008.

[6] C. Castelluccia, S. Jarecki, and G. Tsudik. Secret handshakes from ca-oblivious encryption. In *ASIACRYPT*, pages 293–307, 2004.

[7] COBIS Consortium. COBIS. FP STREP Project IST 004270.

[8] L. Compagna, P. E. Khoury, F. Massacci, R. Thomas, and N. Zannone. How to capture, model, and verify the knowledge of legal, security, and privacy experts: a pattern-based approach. In *ICAIL*, pages 149–153, 2007.

[9] A. Cuevas, P. E. Khoury, L. Gomez, and A. Laube. Security patterns for capturing encryption-based access control to sensor data. *The Second International Conference on Emerging Security*, 2008.

[10] W. Diffie and M. Helman. New directions in cryptography. *IEEE Transactions on Information Society*, 22(6):644–654, november 1976.

[11] Europol, Eurojust, T. Van Cangh, and A. Boujraf. Wp3-cs2: The Eurojust-Europol case study. *at http://www.r4egov.eu/resources*, 2007.

[12] E. Fernandez and R. Pan. A Pattern Language for Security Models. In *In Proc. of PLoP'01*, 2001.

[13] E. B. Fernández, J. Jürjens, N. Yoshioka, and H. Washizaki. Incorporating database systems into a secure software development methodology. *19th International Workshop on Database and Expert Systems Applications*, pages 310–314, 1-5 September 2008, Turin, Italy.

[14] J.-H. Hoepman. Private handshakes. In F. Stajano, C. Meadows, S. Capkun, and T. Moore, editors, *ESAS*, volume 4572 of *Lecture Notes in Computer Science*, pages 31–42. Springer, 2007.

[15] S. Jarecki, J. Kim, and G. Tsudik. Beyond secret handshakes: Affiliation-hiding authenticated key exchange. In *CT-RSA*, pages 352–369, 2008.

[16] A. Mana, C. Rodolph, G. Spanoudakis, V. Lotz, F. Massacci, M. Molideo, and J. S. Lopez-Cobo. *Security Engineering for Ambient Intelligence: A Manifesto*. IGI Publishing, 2007.

[17] C. Meadows. A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party. *Security and Privacy, IEEE Symposium on*, page 134, 1986.

[18] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[19] S. Romanosky, editor. *Security Design Patterns*. 2001.

[20] F. Sanchez-Cid and A. Mana. Patterns for automated management of security and dependability solutions. *1st International Workshop on Secure systems methodologies using patterns (SPattern'07)*, 2007.

[21] F. Sanchez-Cid, A. Munoz, P. El Khoury, and L. Compagna. XACML as a Security and Dependability (S&D) pattern for Access Control in AmI environments. In *Ambient Intelligence Developments - AmI.d*, Sept. 2007.

[22] M. Schumacher. *Security Engineering with Patterns: Origins, Theoretical Models, and New Applications*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.

[23] M. Schumacher, E. Fernandez-Buglioni, D. Hybertson, F. Buschmann, and P. Sommerlad. *Security Patterns : Integrating Security and Systems Engineering (Wiley Software Patterns Series)*. John Wiley & Sons, March 2006.

[24] A. Sorniotti and R. Molva. A Provably Secure Secret Handshake with Dynamic Controlled Matching. *Proc. of 24th International Information Security Conference (IFIP SEC)*, 2009.

[25] G. Spanoudakis, A. Mana Gomez, and K. Spyros, editors. *Security and Dependability for Ambient Intelligence*. Series: Advances in Information Security , Vol. 55, Springer, April 2009.

[26] D. Vergnaud. RSA-Based Secret Handshakes. In *WCC*, pages 252–274, 2005.

[27] S. Xu and M. Yung. k-anonymous secret handshakes with reusable credentials. In *CCS '04: Proceedings of the 11th ACM conference on Computer and communications security*, pages 158–167, New York, NY, USA, 2004. ACM.

[28] J. Yoder and J. Barcalow. Architectural Patterns for Enabling Application Security. In *In Proc. of PLoP'97*, 1997.