Security Patterns for Untraceable Secret Handshakes with optional Revocation

Annett Laube*, Alessandro Sorniotti†‡, Paul El Khoury§‡, Laurent Gomez‡ and Angel Cuevas¶
* *Bern University of Applied Science, Switzerland,*
Email: annett.laube@bfh.ch
† *Institut Eurecom Sophia-Antipolis, France*
‡ *SAP Research Sophia-Antipolis, France*
Email: alessandro.sorniotti@sap.com, laurent.gomez@sap.com
§ *LIRIS University of Claude Bernard Lyon 1, France*
Email: paul.el-khoury@liris.cnrs.fr
¶ *University Carlos III of Madrid, Spain*
Email: acrumin@it.uc3m.es

***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 Untraceable Secret Handshakes, cryptographic protocols that allow two users to mutually verify another's properties without revealing their identity or other sensitive information. 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 from three different domains makes the security solution better understandable for non-security experts and helps to disseminate the security knowledge to application developers.**

***Keywords*-security patterns; secret handshake; cryptographic protocols; mutual authentication;**

I. Introduction

Today's pioneer organizations recognize that performance accelerates when information security is driven into the very framework of business [2]. In the past, security experts developed secure systems and standardized sufficient security solutions to satisfy security requirements for various contexts. Most of these standards provide comprehensive methodologies for specifying, implementing and evaluating security of IT products. Rarely non-specialists are capable of correctly understanding and realizing 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 [3][4].

Application developers transform clients' requirements 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 often 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 [5] and [6] as security patterns, following the SERENITY methodology [4][7]. 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. [5] 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 [8] – to model in a cryptographic protocol the folklore of real handshakes between members of exclusive societies or guilds.

With a secret handshake, two users can simultaneously prove to each other possession of a property, for instance membership to a certain group. The ability to prove and to verify is controlled by a certification authority that issues credentials and matching values respectively. Users are not able to perform a successful handshake without the appropriate credentials and matching values; in addition protocol exchanges should be untraceable and anonymous.

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 security 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 searching into the patterns' library through properties and features.

With the capturing of Untraceable Secret Handshakes

as security patterns, we increase the number of standard security solutions available to application developers. In this paper, we describe not only the secret handshake protocol as in [1], we also discuss variants of the security solution adapted to specific contexts and use cases. The security patterns and their relationships are described in detail. The SERENITY pattern template is extended and complemented by standard UML models. The three scenarios from different domains illustrate the broadness of possible use cases for secret handshakes.

The paper is organized as follows. Work related to secret handshakes is discussed in Section II. In Section III, we overview security patterns and introduce the SERENITY security pattern template. Section IV proposes three scenarios to illustrate the use of secret handshakes in real-world applications. We describe the proposed security solution in detail in Section V and define an abstract model used to capture this security solution as a combination of patterns. Next, Section VI describes the security patterns and integration scheme of the proposed solution. Section VII highlights advantages of the SERENITY pattern approach and points out weaknesses. In Section VIII, we conclude the paper.

II. Related Work

After the introduction of secret handshakes in 2003 by Balfanz et al. [5] 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 with no loss in untraceability) and dynamic matchings (the ability to verify membership for groups different from one's own).

Castelluccia et al. in [9] 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 [10], 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 [11], Hoepman presents a protocol, based on a modified Diffie-Hellman key exchange [12], to test for shared group membership, allowing users to be a member of multiple groups. In [8], Vergnaud presents a secret handshake scheme based on RSA [13]. In [14], 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 et al. 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.

In [16], 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.

Already the Balfanz' original scheme [5] supports revocation, but has a number of drawbacks, for instance the fact that it relies on one-time credentials to achieve untraceability. After this seminal work, many papers have further investigated the subject of secret handshake, considerably advancing the state of the art. The work by Castelluccia et al. [9] has shown how, under some specific requirements (namely CA-obliviousness), secret handshakes can be obtained from PKI-enabled encryption schemes. Other schemes have followed this approach [8][17] offering similar results, albeit with different nuances of unlinkability. Almost all the schemes in this family support revocation of credentials; however the functionalities offered are limited to proving and verifying membership to a common group. In [6], the authors present the first secret handshake with dynamic matching of properties under stringent security requirements. In [18], the revocation support for this kind of secret handshakes is presented.

All the mentioned publications about secret handshakes focus on the cryptographic details of the secret handshake protocols and their formal proofs. Our goal is to capture the common properties and functionalities of secret handshake as general security solution and to make them available for non-security experts. The abstraction from the implementation details allows us to provide a functional view of the needed operations of all involved parties. Our conceptual model and the UML sequence diagrams together with different application scenarios make secret handshakes easier understandable for solution architects and application developers and allow the easy integration of secret handshake in existing or emerging applications.

III. Security Patterns Overview

Research techniques for security patterns are interestingly different from other kinds of research. In software engineering and security engineering innovative results are measured by new solutions brought to market, whereas differently, innovative research in techniques for (*security*) patterns is measured by the successful provision of existing best practices, standards or well proven solutions from experts to lay users. The added value of these techniques for patterns is devoted to the format, validation techniques and means used to promote experts' knowledge to novice security users. Populating a collection of patterns is indeed time-consuming, but once realized the invested effort pays off. To accomplish the security patterns' 'mission', a list of objectives is summarized in four fundamental steps [3]. 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.

In [19], the authors summarize the pattern engineering life cycle, from creation until deployment. In a nutshell this process is presented hereafter as several steps for the creation process, i.e., numbered by 'E', and the deployment one, i.e., numbered by 'A':

- E1 Finding a pair of recurring problem and its corresponding solution from knowledge and/or experiences of software development.
- E2 Writing the found pair with forces in a specific pattern format.
- E3 Reviewing and revising the written pattern.
- E4 Publishing the revised pattern via some public or private resource (WWW, book or paper, ...).
- A1 Recognizing context and security problems in software development.
- A2 Selecting software patterns that are thought to be useful for solving the recognized problems.
- A3 Applying the selected patterns to the target problem.
- A4 Evaluating the application result.

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 [20]. 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 [21]. 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 [23] describe patterns for the most common security models such as Authorization, Role-Based Access Control and Multilevel Security. Recently in [24], 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 were rarely adopted 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 [4][7][25][26] tackles the security patterns objectives. The SERENITY partners presented in [4] the SERENITY model of secure and dependable applications. Moreover, using security patterns they showed how to address, along with the tools provided, the challenge of developing, integrating and dynamically maintaining security mechanisms in open, dynamic, distributed and heterogeneous computing systems.

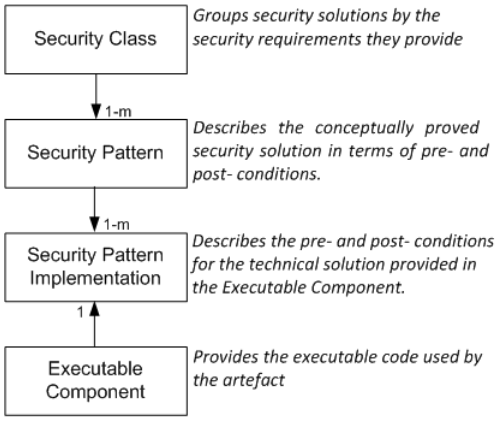

Figure 1. SERENITY Security Artefacts

One of the essential proposals from SERENITY is to provide novice users the SERENITY Security & Dependability pattern package. This package comprises *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 different 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 four artefacts. The description of these artefacts enables selection, adaptation, usage and monitoring at runtime by automated means. The hierarchy is composed of four artefacts (depicted in Figure 1): Security Classes, Security Patterns, Security Implementations and Executable Components. Although this paper emphasized the use of the Security Pattern artefact, in different studies [27][28][29][30] an intuitive and extensive description of all artefacts is presented.

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.

This paper relies on the SERENITY representation of security patterns [4][26] 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 the following:

Problem 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 be found.

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.

Role: The entity applying the pattern is described together with its interactions with other entities from the pattern context.

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 satisfiability 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 provides. 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 used to select suitable patterns.

Consequences: They are the effects (benefits and drawbacks) of the compromise resulting from the application of the pattern's solution. In general, using security solutions implies an increase in cost (economic, more complex mechanisms, etc.).

Variants: This describes variants and possible extensions of the pattern.

Related patterns: They name related patterns, integration schemes and the kind of the relationship (e.g., similarity, dependency, extension).

Often security solutions are too complex to be captured in a single security pattern. Therefore an additional artefact, the *integration scheme (IS)*, was introduced. An IS defines the combination of security patterns and their relationships.

IV. Scenarios

In this section, we want to show how untraceable secret handshakes are used in real-world applications. Our first example is a use case from the EU Project R4eGov [31] for **Mutual Legal Assistance** in international crimes.

Several EU justice forces led by Europol [32] cooperate in order to solve cross-boundary criminal cases (in Figure 2, a workflow example is shown). 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 to authenticate themselves on-the-fly. Both are definitely reluctant to disclose their affiliation and purpose to anybody but the intended recipient.

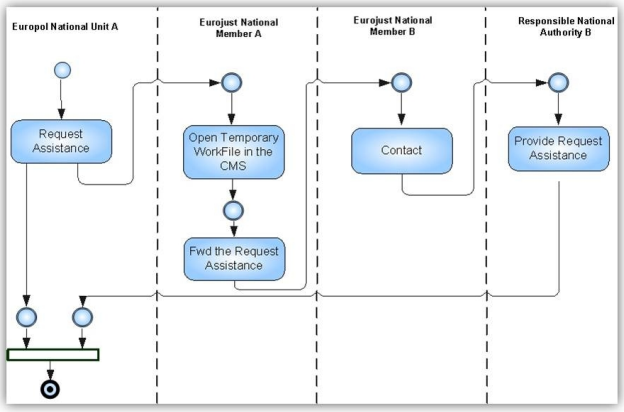

Figure 2. Mutual legal assistance scenario

A scenario from another business domain is the **Incompatible chemicals in proximity** use case from the CoBIs EU project [33]. Let us assume that drums are stored in a warehouse (see Figure 3); 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 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 [6] allow to exchange information of the containing chemicals without revealing them on a wireless channel. Drums with dangerous contents cannot be identified or traced by unauthorized persons.

In the third scenario, we regard **Online Social Networks** (OSNs), like Facebook. A problem, which is particularly felt among social network users, is identity theft and identity spoofing [34]. The root of the problem is that in many OSNs there is little or no verification that a person that joins the social network is really who he or she claims to be. Additionally is the social network users' decision on whether to accept a friendship request based on name, pictures and fragments of text, information that is often easily retrievable elsewhere on the Internet. A viable solution consists on users creating ad-hoc, trusted groups outside of the social network, issuing group membership credentials and presenting such credentials upon friendship invitations within the social network. A natural evolution of the aforementioned trusted friend groups are **Secret Interest Groups (SIGs)** [35], user-created groups with particular attention to confidential or simply privacy-sensitive topics. Indeed users of online

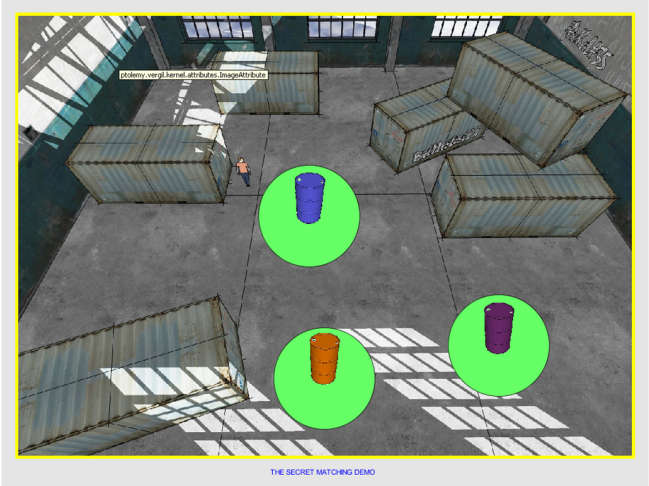

Figure 3. Incompatible chemicals in proximity scenario

social networks are often also exchanging personal and sensitive material; moreover, OSNs are more and more the theater of political, religious debate, often used as means to exchange confidential material that cannot go through official channels. Secret handshakes [5] are the main pillar to fulfill the operational and security requirements of a generic SIG framework.

Often it is desirable to support **revocation** for secret handshakes. In the SIG use case, revocation is required when either a SIG member got his membership token stolen or when he no longer qualifies for membership. Since SIG membership credentials can be used directly to authenticate to another SIG member a *reactive revocation* approach is required, that singles out revoked credentials based on a revocation list. In the scenario of assistance in international crimes, the credentials of the operating officers to authenticate should be automatically revoked when the criminal case was closed. *Proactive revocation* techniques are based on time-bound credentials, which have to be updated periodically, can be used in this case.

V. Solution Description

A *Secret Handshake*, first introduced in [5], 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 to verify is strictly controlled by a certification authority that issues property credentials and matching values. Users are not able to perform a successful handshake without the appropriate credentials and matching values; in addition protocol exchanges have to be untraceable and anonymous.

We present a pattern for Untraceable Secret Handshakes with proof of group membership as described in [5] or [6]: 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.

The secret handshake requires an *initialization phase* followed by a *matching phase*, which can be repeated several times.

A. Initialization phase

A secret handshake is performed between two parties, in the following also called **users**. To carry out a secret handshake each user needs a property credential and matching values. A **property 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. To verify a certain property the identity of the user and the related context are examined. This operation is described in the pattern *Property certification* and can be an offline step.

The CA can be a single person or organization, like Europol, the European Law Enforcement Organisation, in the Mutual Legal Assistance scenario. Europol assists the authorities in the EU Member States in preventing and combating terrorism, and other serious forms of international organized crime. Europol can certify the involvement of a justice force in a specific criminal case. In the third scenario, a non-empty group of so called SIG managers is responsible to verify the group membership in an offline process and to manage the property credentials, in this case called *membership credentials*.

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 described in the security pattern *Property Certification*. The policy with the relationships between property credentials and matching values has to be defined beforehand.

Figure 4 depicts an abstract model highlighting input, output and entities specific for this phase. The data flow between the CA and the user helps to understand the proposed solution. The security pattern is shown as rounded rectangle including the operations the applying entity has to perform. The exchanged data and documents (e.g., credentials and policies) are depicted together with the flow to and from the manipulating operation.

Like described in Section IV, revocation support can be required for a secret handshake scenario. Depending on the desired revocation technique (reactive or proactive) the properties credentials have to be generated differently. The cryptographic details for generating credentials with

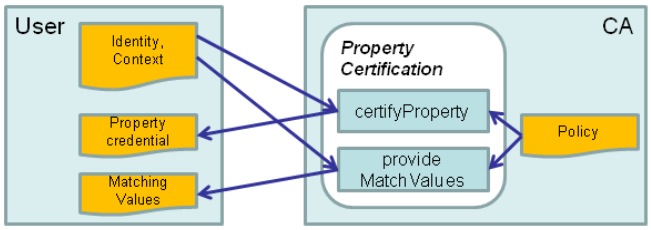

Figure 4. Conceptual Model - Initialization phase

identification handles for reactive revocation are described in [18]. The proposed scheme there, offers a solution that on the one hand assures that the protocol messages are untraceable, and on the other offers a credential tagging to single out the revoked ones.

B. Matching phase

The secret handshake itself is carried out between two users and can be repeated infinitely. It consists of two mandatory parts: the secure match of properties and the proof that both parties possess the same key after the matching. The revocation support can be added optionally when needed. The relationships between the security patterns and the data flow between the entities are pictured in Figure 5.

The secure match is initiated by one of the users, e.g., user A. User A sends an internal state (state A), e.g., a nonce, to user B. User B replies with another state (state B) 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., [6] for details about the used cryptographic algorithms). 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.

When reactive revocation is supported by the property credentials (i.e., they contain an *identification handle*) and the CA maintains a list of revoked credentials (represented by so called **revocation handles**) the secure match can be extended. Before user A matches the received hidden credential from user B, user A checks if the received credential B is invalid. Invalid means that the revocation handle computed from the received credential matches one of the revocation handles from the publicly available revocation list. This behaviour is caught by the pattern *Credential Revocation*.

After a successful matching both parties, user A and B, own a secret, for instance a key (see key A and key B in Figure 5), 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, PoK)* can be used. The parties exchange encrypted information to prove their knowledge without disclosing the key directly.[1] User A sends a challenge to user B. The challenge 1 contains an internal state, e.g., a random number, encrypted with the key obtained from the secure match (key A). User B replies with challenge 2: he uses his key B from the secure match to decrypt the challenge 1, modifies the result (e.g., he increments the number by 1) and sends this encrypted with his key B. User A can now verify that user B has obtained the same key while decrypting challenge 2 with his key and inversing the operation from user B (e.g., decrementing the result by 1). If the result is identical to the state used in challenge 1, user A has proved that B has an identical symmetric key. To complete the protocol, user A has to reply challenge 2. User A decrypts challenge 2, modifies the result in the agreed way and sends back the encrypted result. User B can now perform the verification on his side.

VI. Security Patterns and Integration Scheme

We identified the following four security patterns: *Property Certification*, *Secure Match*, *Credential Revocation* 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.

A. Property Certification Pattern

Problem and context: The secret handshake 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.

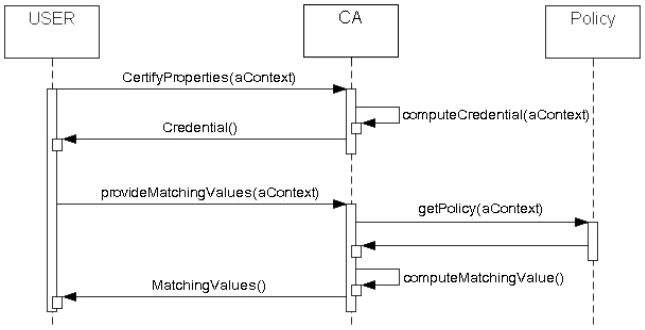

Figure 6. UML diagram - Property Certification

Solution: The pattern defines the following operations. In Figure 6, the interaction between the entities is shown.

- **certifyProperty**: The input of this operation is the application context of the handshake. The context contains

[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.

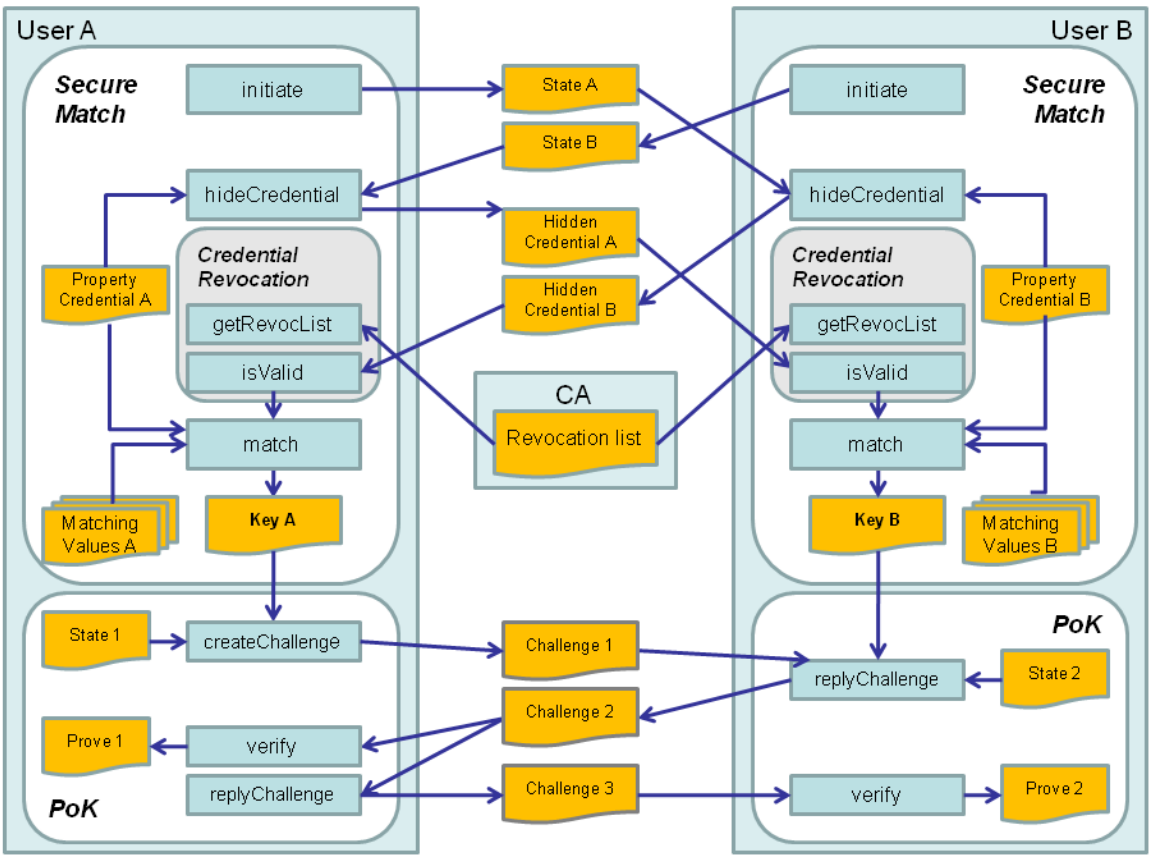

Figure 5. Conceptual Model - Matching phase

all information needed to decide about the possession of a predefined property, e.g., the user's identity and the process where he is involved in. The possession of the user's property is verified in the given context and if this was successful, a credential (resp. *property credential*) representing the property is returned.

- **provideMatchingValues**: The input of this operation is the application context (including user's identity) of the handshake. According to the policy, the operation returns a set of *Matching Values*.

Roles: The pattern is applied by a Certification Authority, which grants property credentials to users and provides them with 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 have been defined.
- Communication channels are secured.

Properties: Certification, Policy Enforcement

Features: Revocation Support (optional)

Consequences: The issued property credentials and matching values have been kept secret by the users and stored in a safe place.

Variants: According to requirements of revocation support, the created credentials have to carry an *identification handle* to support reactive revocation (see pattern *Credential Revocation* in Section VI-C) or to be time-bound for proactive revocation.

Relationships: The pattern realizes the initialization phase of a secret handshake (IS *Untraceable Secret Matching*). The second phase - the matching phase - of the secret handshake is described by the patterns *Secure Match* and *Mutual Key PoK* (see Section VI-B and VI-D).

B. Secure Match Pattern

Problem and context: A user wants to exchange secretly credentials with another user in order to verify that the other party possesses a matching property.

Solution: The pattern defines the following operations. In Figure 7, the protocol between the two 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 to the other party.
- **match**: Input of the operation is the received hidden credential from the other party, the owned property cre-

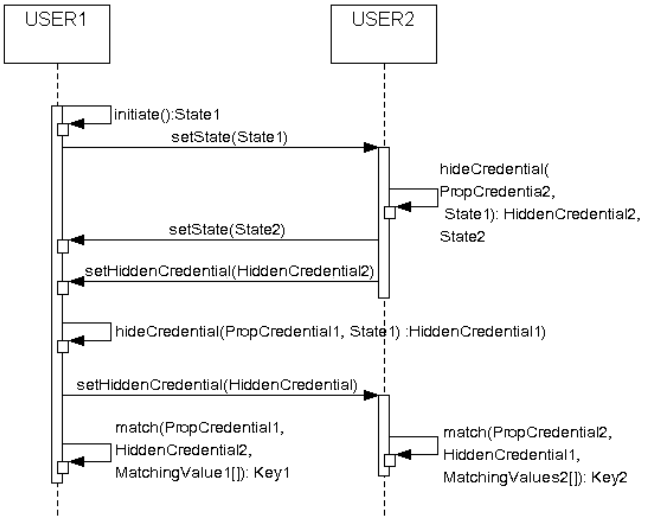

Figure 7. UML diagram - Secure Match

dential 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.

Roles: The pattern is applied by 2 parties (users) who want to authenticate secretly.

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: Authentication, Property Validation

Features: Untraceability, Key establishment, Fairness, Revocation Support (optional)

Consequences: None

Variants: The pattern *Credential Revocation* can be integrated to support reactive revocation.

Related patterns: The pattern realizes the matching phase of a secret handshake (IS *Untraceable Secret Matching*). The first phase - the initialization phase - is described by the pattern *Property Certification*. The pattern can be extended by pattern *Credential Revocation*.

C. Credential Revocation Pattern

Problem and context: It is possible that a (property) credential becomes invalid (e.g., the credential was compromised or a user has to leave a certain group). The user performing a secret handshake has to know, if the credential of the other party is valid in order to refuse any interactions with the concerned users.

Solution: The pattern defines the following operations.

- **getRevocationList**: Retrieves the list of revocation handles from the CA.
- **isValid**: The function computes first the revocation handle from the received hidden credential from the other party (details of the computation are in [18]). In a second step, it is verified that this handle is not part of the revocation list.

Roles: The pattern is applied by a user who wants to verify if a credential is valid and not revoked.

Pre-Conditions:

- The chosen secret handshake protocol supports revocation, that means the property credentials granted by the CA contain *identification handles*.
- The CA maintains a publicly available list of revocation handles.
- Compromised property credentials have to be announced to the CA.
- Dependent on the application context, the CA has to verify regularly that all group members are still qualified for group membership and otherwise to add the non-qualified members to the revocation list.

Properties: Revocation support for secret handshakes

Features: Anonymity and untraceability for valid credentials, traceability for revoked credentials

Consequences: Additional communication effort for the user to retrieve the revocation list from the CA before performing a secret handshake. Additional computation effort for the user to perform the operation isValid (computational cost $O(\frac{n}{2})$ in average, worst case $O(n)$).

Variants: None

Related patterns: The pattern is an optional part of the *Secure Match* pattern. The pattern is functional similar to the security pattern *Certificate revocation* described in [36]. The difference lays in the revoked objects (certificates vs. property credentials).

D. Mutual Key Proof of Knowledge Pattern

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

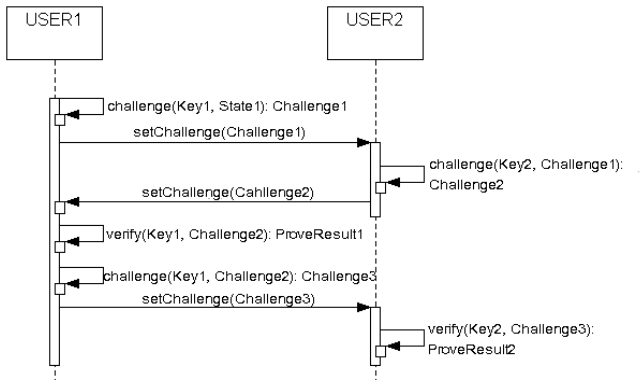

Figure 8. UML diagram - Mutual Key PoK

Solution: The pattern defines the following operations. In Figure 8, the protocol between the two parties is shown.

- **createChallenge**: A challenge is sent to the other party, containing e.g., an internal state (random number) encrypted with the owned key.
- **replyChallenge**: A received challenge is decrypted with the own key. An agreed operation (e.g., increment by 1) is applied to the decrypted value. The result is encrypted and sent back to the other party.
- **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.

Roles: The pattern is applied by 2 parties (users) who want to verify mutually if they possess an identical 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.

Variants: None

Related patterns: Can be used in the matching phase of a secret handshake (IS *Untraceable Secret Matching*).

E. 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 I. We used the keyword $roles$ of the SERENITY security patterns' language [4] in order to separate between the two users of the protocol using the same security pattern.

The IS contains the security property **Mutual Authentication**. That means that the complex security solution provides a mechanism that allows two parties authenticating each other based on property credentials and corresponding matching references controlled by a certification authority.

Additional security features of the IS are:

- **Untraceability**: Consider an adversary with valid property credentials and matching references and able to perform a secret handshake with legitimate users. His goal is – given any two disguised credentials - to trace them to having been generated from the same credential, so as to prove possession of the same property and at the same time to the same group. The attacker cannot decide whether there is a property that both credentials can be matched to.
- **Impersonation Resistance**: It is computationally infeasible for an attacker with valid credentials and matching references, to impersonate a user owning a given property credential, which the attacker does not dispose of and did not steal.
- **Detector Resistance**: An adversary cannot verify the presence of a property of his choice without owning the corresponding matching references. That means that an adversary with valid credentials cannot find out while performing a secret handshake if the other party belongs to a group where he is not a member of.
- **Anonymity**: The identity of the users applying the secret handshake is not disclosed until the protocol was finished and both parties could match the received credentials with their matching references. The parties performing the secret handshake stay anonymous until the protocol was finished with a successful match on both sides. In cases of unfinished protocols or unsuccessful matches on one or both sides, both parties have no knowledge about the other parties' identity or properties.

Table I
IS OPERATIONS

```
Property Certification Pattern ← CA
Secure Match Pattern ← SMP
Credential Revocation Pattern ← CRP
Mutual Key PoK Pattern ← MKPP
SMP ← roles: User1 and User2
CRP ← roles: User1 and User2
MKPP ← roles: User1 and User2
—— Initialization phase ——
CA.certifyProperty(in:Identity1, in:Context1, out:PropCredential1);
CA.certifyProperty(in:Identity2, in:Context2, out:PropCredential2);
CA.provideMatchingValues(in:Identity1, in:Context1, out:MatchValue1[]);
CA.provideMatchingValues(in:Identity2, in:Context2, out:MatchValue2[]);

—— Matching phase ——
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.CRP.getRevocationList(out:RevocationList);
User1.CRP.isValid(in:HiddenCredential2, in:RevocationList,
    out:BooleanResult1);
if (BooleanResult1 == true) {
    User1.SMP.match(in:PropCredential1, in:HiddenCredential2,
        in:MatchValue1[], out:Key1);
}
User2.CRP.getRevocationList(out:RevocationList);
User2.CRP.isValid(in:HiddenCredential1, in:RevocationList,
    out:BooleanResult2);
if (BooleanResult2 == true) {
    User2.SMP.match(in:PropCredential2, in:HiddenCredential1,
        in:MatchValue2[], out:Key2);
}
if (BooleanResult1 == true && BooleanResult2 == true) {
    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)
}
```

- **Resistance to replay attacks**: An adversary cannot successfully perform a secret handshake by repeating parts of the protocol that was eavesdropped from a successful handshake between two parties. Nor he can learn something about the identity or properties of the other party.
- **Revocation support** (optional): Property credentials can be *reactively* revoked when needed by adding them to a public available revocation list. *Proactive* revocation is possible with the help of time-bound credentials.

All listed features can be proved formally under the assumption that the Bilinear Decisional Diffie-Hellman (BDDH) problem is hard (details in [6]).

To provide the complex solution all pre-conditions defined in the embedded security patterns have to be considered before the IS can be applied.

VII. Analysis of SERENITY Security Patterns

While applying the SERENITY methodology to the new area of cryptographic protocols, some advantages but also some weaknesses of the pattern approach became evident.

The template used for the SERENITY security patterns allows to describe the security solution in an structured way, identifying the applying entities, their relationships and interactions. This goes far beyond the usual used natural language description, applied e.g., in [36] or [37]. Especially the description of the pattern solution as sequential flow of operations is more explicit and closer to a possible implementation. Therefore it is easier understandable for application developers used to read and interpret design patterns. The operations are the link to the other artefacts of the SERENITY methodology, like *Security Pattern Implementation* and *Executable Components*, which describe the detailed technical solution and provide the executable code.

In our security patterns, an operation consists often of two types of activities:

- **computation activities** like cryptographic computations or policy evaluations that are executed by the entity applying the pattern,
- **communication activities** where input (like parameters) is received from a partner entity or where the result of the computation is sent out.

A good example is the operation *certifyProperty* in the pattern *Property certification*. The CA applying the pattern receives the application context and identity from the user who wants to get a property certified. After successful validation of the user's context, the CA sends the property credential back to this user.

This shows an improvement possibility of the SERENITY pattern approach. In the pattern description, only the applying actor is described. This is not sufficient, when protocols, simple interactions or data exchanges are part of the security solution. In this paper, we decided to use UML sequence diagrams to correlate the pattern operations with the interactions of the different entities.

We could identify a related issue in the integration scheme, describing the security solution as composition of several security patterns. The IS operational flow (see Table I) is not descriptive enough to clearly describe the information exchange between the involved entity and needs more details than a simple sequence of operations. The operations define a number of input and output parameters, but there is no means to differentiate between a simple in-/output that is internal to the applying entities or an information exchange with another entity.

Standard modeling languages, like UML or UMLSec, can be used to describe the complex data flow and interactions in the protocols. This has also the advantage that these languages are well-known by application developers and the diagrams, e.g., sequence diagrams, are easy to understand. Information about activities that can be done in parallel, repeated, omitted or those relying on secure channels can be easily added.

We propose a *conceptional model* (see Figures 4 and 5) to visualize the data flow in the security solution. It is similar to an UML activity diagram and shows the involved entities applying the different security patterns and pattern operations as well as the exchanged information.

Another weakness of the security pattern approach is the pattern selection process. Can an application developer or architect responsible to implement one of the scenarios from Section IV 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, security properties and features in a way non-security experts understand.

The problem of the definition of security properties and features is two-edged. The increasing number of security solutions as patterns demands a detailed classification of the security properties and features in order to be able to select the right pattern for a given problem. On the other hand, we have the non-security expert confronted with a huge set of terms related to security. In order to enable these people to use the security properties to select the right pattern, we need a clear, unique and understandable definition of each of them. There should be a catalogue of security properties, which allows with the help of a taxonomy to navigate in the property space.

VIII. Conclusion

In this paper, we described the Untraceable Secret Handshake protocol as SERENITY security pattern. To our knowledge, was this one of the first attempts to describe cryptographic schemes and protocols as formal security patterns

([1] provided the first definition of the Secret Handshake pattern).

In spite of the discovered weakness of the security pattern approach in general, the abstract description of the secret handshake protocols independent of the real algorithms helps to spread the knowledge in the security community and also among application developers and architects. Secret handshakes and secure matching are in general difficult to understand for non-cryptographic experts. The security pattern approach makes them available for people without deep cryptographic knowledge.

Following the SERENITY methodology, each security pattern has one or more possible implementations (*Security Pattern Implementations* and *Executable Components*), which give concrete examples of the security solution in different environments and show their feasibility.

In our case, we implemented two of the scenarios described in Section IV. A first solution is based on the simulation framework Ptolemy II [38] and implements the *Incompatible chemicals* scenario. The simulated sensor nodes are initialized with the properties credentials for the liquid in the attached drum and the matching value for all reactive liquids. When a sensor intersects the wireless range of a second sensor, the sensors perform the secret handshake and create an alert when necessary.

The second solution is a Java implementation of the *Secret Interest Groups* in the framework for Facebook motivated by its extreme popularity. The intended use is the following: a SIG member runs the java http proxy, which intercepts only requests toward Facebook servers. The proxy modifies requests and responses, running the secret handshake protocol upon membership invitation and chat events; notification of the success or failure of the protocol is provided to the user through modifications of the html that is displayed in the browser. Further details about the implementation are in [35].

Additionally, we developed in the context of the WASP project [39] a solution based on web services to mutually authenticate mobile Wireless Sensor Network gateways with different backend applications.

The three different implementations proved the feasibility of the developed security pattern for secret handshakes. Despite the different development environments, it was possible to follow the SERENITY approach. In each prototype, we could identify the described security patterns and their interactions.

The different developed solutions show also that mutual authentication with untraceability and attacker resistance is a recurring security problem that arises in different contexts. The security patterns captured in this paper represent a well-proven solution based on the cryptographic schemes of secret handshakes.

REFERENCES

[1] A. Cuevas, P. El Khoury, L. Gomez, A. Laube, and A. Sorniotti, "A Security Pattern for Untraceable Secret Handshakes," in *Third International Conference on Emerging Security Information, Systems and Technologies, SECURWARE '09*, Mar. 2009.

[2] P. El Khoury, M.-S. Hacid, S. S. Kumar, and E. Coquery, *A Study on Recent Trends on Integration of Security Mechanisms*, Mar. 2009, ch. Advances in Data Management, special volume of Studies in Computational Intelligence.

[3] 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.

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

[5] 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*, 2003, pp. 180–196.

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

[7] 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.

[8] D. Vergnaud, "RSA-Based Secret Handshakes," in *WCC*, 2005, pp. 252–274.

[9] C. Castelluccia, S. Jarecki, and G. Tsudik, "Secret handshakes from ca-oblivious encryption," in *ASIACRYPT*, 2004, pp. 293–307.

[10] 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*, p. 134, 1986.

[11] J.-H. Hoepman, "Private handshakes." in *ESAS*, ser. Lecture Notes in Computer Science, F. Stajano, C. Meadows, S. Capkun, and T. Moore, Eds., vol. 4572. Springer, 2007, pp. 31–42.

[12] W. Diffie and M. Helman, "New directions in cryptography," *IEEE Transactions on Information Society*, vol. 22, no. 6, pp. 644–654, november 1976.

[13] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[14] 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*. New York, NY, USA: ACM, 2004, pp. 158–167.

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

[16] G. Ateniese, M. Blanton, and J. Kirsch, "Secret handshakes with dynamic and fuzzy matching," in *Network and Distributed System Security Symposuim*. The Internet Society, 02 2007, pp. 159–177, cERIAS TR 2007-24.

[17] S. Jarecki and X. Liu, "Unlinkable secret handshakes and key-private group key management schemes," in *ACNS'07*, 2007, pp. 270–287.

[18] A. Sorniotti and R. Molva, "Secret handshakes with revocation support," in *ICISC 2009, 12th International Conference on Information Security and Cryptology, Seoul, Korea*, 12 2009.

[19] N. Yoshioka, H. Washizaki, and K. Maruyama, "A survey on security patterns," *Progress in Informatics*, vol. No. 5, pp. 35–47, 2008.

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

[21] S. Romanosky, Ed., *Security Design Patterns*, 2001.

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

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

[24] 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*, pp. 310–314, 1-5 September 2008, Turin, Italy.

[25] L. Compagna, P. El 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*, 2007, pp. 149–153.

[26] 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*, Sep. 2007.

[27] 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.

[28] A. Benameur, P. El 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. Vol. 55, 2009.

[29] A. Cuevas, P. El 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.

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

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

[32] [Online]. Available: http://www.europol.europa.eu/

[33] COBIS Consortium, "COBIS. FP STREP Project IST 004270," 2005. [Online]. Available: www.cobis-online.de

[34] [Online]. Available: http://chris.pirillo.com/pownce-social-networks-arent-identity-networks/

[35] A. Sorniotti and R. Molva, "Secret interest groups (SIGs) in social networks with an implementation on Facebook," in *SAC 2010, 25th ACM Symposium On Applied Computing, March 22-26, 2010, Sierre, Switzerland*, 2010.

[36] S. Lehtonen and J. Pärssinen, "A pattern language for cryptographic key management," in *EuroPLoP*, 2002.

[37] C. Steel, N. Ramesh, and L. Ray, *Core Security Patterns: Practices and Strategies for J2EE, Web Services, and Identity Management*. Upper Saddle River: Prentice Hall PTR, 2005.

[38] 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*. University of California at Berkeley, 2004, vol. 1.

[39] WASP, "WASP (Wirelessly Accessible Sensor Populations), IST 034963," 2006. [Online]. Available: www.wasp-project.org