



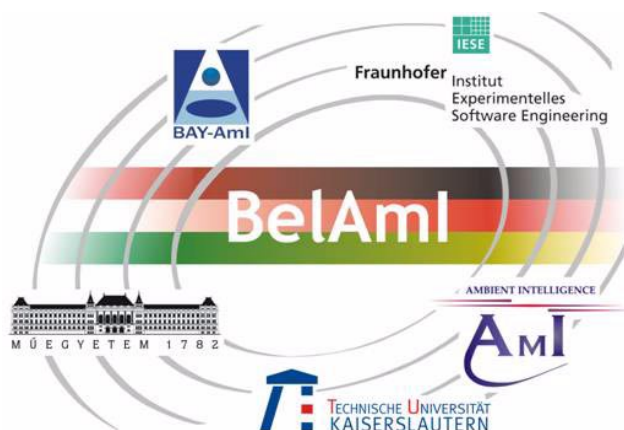
Fraunhofer Institut
Experimentelles
Software Engineering

Security Design Patterns for Ambient Systems

BelAml Project Deliverable D3.6.1

Authors:

Reinhard Schwarz
Kai Simon



BelAml Report No. 009.06/E
IESE Report No. 115.06/E

Date: October 2006
Version: 1.0
Status: final
Classification: public

A publication by Fraunhofer IESE

Fraunhofer IESE is an institute of the Fraunhofer Gesellschaft. The institute transfers innovative software development techniques, methods and tools into industrial practice, assists companies in building software competencies customized to their needs, and helps them to establish a competitive market position.

Fraunhofer IESE is directed by
Prof. Dr. Dieter Rombach (Executive Director)
Prof. Dr. Peter Liggesmeyer (Director)
Fraunhofer-Platz 1
67663 Kaiserslautern

© 2006 Fraunhofer IESE and TU Kaiserslautern.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means including, without limitation, photocopying, recording, or otherwise, without the prior written permission of the publisher. Written permission is not needed if this publication is distributed for non-commercial purposes.

Document Change History

Version	Date	Author	Comment
1.0	October 2006	R. Schwarz, K. Simon	Initial version

Abstract

Software design patterns have been gaining popularity since their introduction by the seminal work of the so-called »Gang of Four« (Gamma, Johnson, Helm, and Vlissides [8]). A design pattern describes a recurring design problem that arises in specific design contexts, and it presents a well-proven generic solution for it. Patterns proved their worth in conveying object-oriented design principles to the software community. The success of general design patterns inspired software engineers to apply the pattern approach to the domain of software security engineering to provide prefabricated building blocks for typical security problems.

This report surveys existing software security design patterns proposed in the literature, trying to identify candidate patterns that are applicable to ambient systems design. For each candidate pattern we discuss the pros and cons of its application, emphasizing potential uses in scenarios in the context of our BelAml demonstrator.

This report is Deliverable D3.6.1 according to the BelAml project plan.

Keywords

IT security, system security engineering, security patterns, design patterns, ambient intelligence, pervasive computing, ubiquitous computing, privacy, survey, BelAml

Preface

Inspired by the success of object-oriented software design patterns, research began to explore the application of patterns in other software engineering disciplines, not only restricted to the design phase of the software development lifecycle. This »patterns everywhere« movement extended the scope of software patterns to all aspects of software systems design and implementation, ranging from core design issues such as human-computer interaction to management aspects such as resource planning.

Recently, security was identified as another hot topic for patterns, and several conferences, interest groups, research papers and text books specifically address the broader issues of software security patterns, see [12, 13, 20, 26, 27] for typical examples. Despite the considerable body of literature on the topic of software (security) patterns, there is only a small intersection of security pattern research and ambient/ubicom/pervasive systems research: Presently, the pattern community seems to address mostly information systems, neglecting the security needs of the embedded systems domain.

In this report, we aim at the intersection between software security design patterns and ambient systems design patterns. Due to the scarcity of available material, our survey uses a broad definition of »ambient system«; for the purpose of our study, we treated the related terms »ubiquitous systems« and »pervasive systems« as synonyms.

We used the BelAml »assisted living« demonstrator project to provide real-world examples for the application of security patterns. More specifically, we refer to a snapshot of the demonstrator components and their respective security requirements as described in [28]. We took our demonstrator scenarios to reflect on the suitability of the various security design patterns proposed in the literature. Some of the security solutions devised for the BelAml demonstrator inspired new or modified patterns.

Of course our demonstrator can only support a limited number of security-related scenarios. Thus, only a subset of patterns was suited for direct application in the BelAml »assisted living« context. Other patterns are more characteristic for domains such as »mobile location-based services« or »vehicular ad-hoc networks« and other Aml technologies. Nevertheless, these patterns were included in our survey.

Inhaltsverzeichnis

Document Change History	v	
Abstract	vii	
Preface	ix	
List of Security Design Patterns	xiii	
1	Covering Aml Security Needs with Design Patterns	1
1.1	Design patterns specific to ambient systems	2
1.2	Pattern format used in our survey	2
1.3	Applying software security design patterns	4
2	Authentication, Authorization, and Access Control	5
2.1	Peculiarities of authentication, authorization, and access control	5
2.2	Design patterns for authentication, authorization, access control	6
2.2.1	<i>Pattern »Callback Authentication« (AAA_CBA)</i>	6
2.2.2	<i>Pattern »Location-dependent Access« (AAA_LDA)</i>	8
2.2.3	<i>Pattern »Emergency Access« (AAA_EMA)</i>	10
2.2.4	<i>Pattern »Slave Imprinting« (AAA_SLI)</i>	13
2.2.5	<i>Pattern »Plutocratic Access Control« (AAA_PAC)</i>	15
3	Privacy and Anonymity	17
3.1	Peculiarities of privacy	17
3.2	General privacy principles	18

3.3	Design patterns promoting privacy	20
3.3.1	<i>Pattern »Privacy Sensitive Architecture« (PRI_PSA)</i>	20
3.3.2	<i>Pattern »Privacy Enhancing Technology« (PRI_PET)</i>	22
3.3.3	<i>Pattern »Blurred Personal Data« (PRI_BPD)</i>	24
3.3.4	<i>Pattern »Partial Identification« (PRI_PAI)</i>	26
3.3.5	<i>Pattern »Beacon« (PRI_BCN)</i>	26
3.3.6	<i>Pattern »Trusted Personal Device« (PRI_TPD)</i>	30
3.3.7	<i>Pattern »Physical Privacy Zone« (PRI_PPZ)</i>	32
3.3.8	<i>Pattern »Invisibility Mode« (PRI_INV)</i>	34
3.3.9	<i>Pattern »Privacy Feedback« (PRI_PFB)</i>	36
3.3.10	<i>Pattern »Limited Data Retention« (PRI_LDR)</i>	38
4	Establishing Trust	41
4.1	Factors contributing to trust	41
4.2	Design patterns that promote trust	43
4.2.1	<i>Pattern »Trust Enhancing Architecture« (TR_TEA)</i>	43
5	Summary and Outlook	45
	References	47
	List of Abbreviations	51

List of Security Design Patterns

Callback Authentication	AAA_CBA	7
Location-dependent Access	AAA_LDA	9
Emergency Access	AAA_EMA	11
Slave Imprinting	AAA_SLI	14
Plutocratic Access Control	AAA_PAC	16
Privacy Sensitive Architecture	PRI_PSA	21
Privacy Enhancing Technology	PRI_PET	23
Blurred Personal Data	PRI_BPD	25
Partial Identification	PRI_PAI	27
Beacon	PRI_BCEN	29
Trusted Personal Device	PRI_TPD	31
Physical Privacy Zone	PRI_PPZ	33
Invisibility Mode	PRI_INV	35
Privacy Feedback	PRI_PFB	37
Limited Data Retention	PRI_LDR	39
Trust Enhancing Architecture	TR_TEA	44

1 Covering Aml Security Needs with Design Patterns

Design patterns have been very influential in object-oriented software engineering since the appearance of the seminal publication by the so-called Gang of Four in 1995 [8]. Design patterns transfer the metaphor of a proven, successful architectural pattern for constructing buildings to the realm of software. They promote reuse — and also software quality.

Extending the metaphor one step further, »security patterns« have been suggested as prototypes for security solutions. One motivation is to provide better security with less effort — and more reliably and predictably; the other is to leverage expert knowledge to a broader audience. At a minimum security patterns are expected to provide useful insight into recurring security problems and their solutions.

There are several problems and open research issues with existing pattern collections, though:

- Security patterns are provided at different levels of abstractions and for different needs. Different people count different artefacts at different levels of formality as a pattern: formal textual descriptions versus UML-like specifications with well-defined formal semantics; structural patterns describing architecture versus procedural patterns describing system lifecycle processes.
- Many security properties do not easily fit into a fixed pattern that could be depicted as, for example, a UML component, but emerge from the system design as a whole.
- Security is about correct behavior and state in every detail, not about »roughly correct at large«: Doesn't that contradict the template approach of generalized patterns? Satisfying the pattern only superficially, at macroscopic level may give us a deceptive feeling of security.
- Many patterns are too abstract, providing little tangible guidance in project context; others are almost too detailed, limiting their portability. Consensus has not been reached yet what the suitable level of abstraction should be.

Furthermore, research in security patterns is still rather software-centric, dealing mostly with the classical problems of information security. We still lack patterns

that address the integrated systems view of software, hardware, and environment, that is typical for security in the realm of embedded systems design.

1.1 Design patterns specific to ambient systems

In spite of the relative immaturity of software design patterns in general, and of Aml security design patterns in particular, our survey tries to provide an account of the state-of-the-art in security design patterns applicable to Aml systems.

The design patterns that are most specific for ambient systems center around aspects of privacy, anonymity, trust, and transparency. This is not surprising, given that the Aml paradigm postulates mostly invisible, autonomous systems that act behind the scenes, sometimes without explicit user mandate. Apparently, one of the biggest challenges is how to design an Aml system that makes its users feel comfortable, safe, and secure.

We excluded Aml design patterns from our survey if they were not related to security. The reader who is interested in more general design patterns specific to ambient/ubicom/pervasive systems is referred to the work of the *Group for User Interface Research* (GUIR) at the University of California at Berkeley [5], specifically to their compilation of design patterns for ubicomp systems that can be downloaded from the project's home page. The impact of ubicomp patterns on the design of digital home applications has been studied in [25]. Both [5] and [25] provide empirical evidence for the benefits of design patterns.

To the best of our knowledge, the most comprehensive overview of general security patterns — not necessarily *design* patterns or patterns specific to ambient/ubicom/pervasive systems — is currently provided by Schumacher et al. in [27]. A collection of Java security patterns is presented in [33].

1.2 Pattern format used in our survey

Several collections of security patterns are publicly available for download by various organizations and research groups, see for example [1, 5, 12, 26]. The various authors proposed slightly different pattern formats to present their pattern compilations. By comparing the suggested variants, we selected the following core attributes for our pattern descriptions:

- Pattern Name — a descriptive name followed by a unique label
- Abstract — a brief summary of purpose and intent of the pattern
- Synonyms — other names for the pattern found in the literature

- Background — the context of the underlying security problem
- Problem — the specific security problem addressed by the pattern
- Solution — a high-level description of how the pattern solves the problem
- Issues — hints and caveats concerning the application of the pattern
- Examples — known uses of the pattern
- Static structure — hints on architectural peculiarities
- Runtime dynamics — hints on runtime peculiarities, proper operation etc.
- Consequences — the potential impact of the pattern on other functional or non-functional properties (with focus on primary effects), in particular with respect to
 - accountability, availability, confidentiality, integrity
 - maintainability, usability
 - performance, cost
- Related Patterns — links to other useful patterns in this context
- Variants — potential variations of the pattern’s key idea
- References — citations in the literature related to the pattern

For most patterns only a subset of these attributes is relevant. To avoid clutter, we omitted irrelevant attributes in our pattern characterizations. Furthermore, we divided the set of attributes into two subsets:

- The most important pattern properties (Name, Abstract, Background, Problem, Solution, Issues, Examples, and Related Patterns) are combined to a one-page synopsis in tabular form.
- The remaining properties are excluded from the one-page characteristics, but only appear as subsections of the accompanying textual description.

We excluded the latter attributes mainly for two reasons: Either they turned out to be irrelevant for most of our patterns, or — if relevant — they typically warrant a lengthy discussion that is unsuitable for our compact tabular pattern characterization.

1.3 Applying software security design patterns

The first step in applying security design patterns is to identify the system's or application's relevant information assets. Next, the designer should highlight for each asset the asset's security, privacy, data protection, and trust needs. These needs provide a first indication for the classes of patterns that are potentially relevant.

For each class identified, the software designer can browse the corresponding sections of our pattern survey for applicable security patterns. Before selecting a pattern, the designer should read the »Issues« and »Consequences« descriptions to check for potential contraindications.

If the pattern actually fits the designer's needs, it must be instantiated in the local context of the system under design. At this stage, annotations on »Static structure« and »Runtime dynamics« may provide additional guidance for the implementation of the pattern.

2 Authentication, Authorization, and Access Control

This chapter presents security design patterns for the proper verification of user identities, and for the assignment of access permissions based on authorization rules for authenticated or unauthenticated system users. We avoid the usual standard patterns that have no particular relevance for ambient systems but are widely employed in traditional information systems. Rather, we concentrate on those patterns that offer a contribution to a security problem characteristic of Aml environments.

2.1 Peculiarities of authentication, authorization, and access control

The needs of authentication, authorization, and access control (AAA) generally conflict with seamless user–system interaction. The Aml paradigm exacerbates this problem because it specifically postulates casual use and ad-hoc interaction, without laborious login procedures. Furthermore, the target group for Aml technology are »computer illiterates« such as elderly people, who can hardly be expected to grasp all subtleties of AAA.

Therefore, Aml system designers have to look for unintrusive and intuitive means to authenticate and empower legitimate users — and to identify and exclude malevolent individuals or hostile devices. Unfortunately, user-friendly solutions are restricted by several trade-offs:

- Seamless AAA should best be left to personalized devices acting on behalf of their owners, but Aml solution must be inexpensive, leaving little room for designs based on special user equipment.
- Strong authentication requires trust, but establishing trust relations takes time and needs preparation.¹ Lengthy preparation, however, is inconsistent with ad-hoc participation.
- For strong authentication there is (presently) no alternative to cryptography, which is far from simple or intuitive.

¹ Note that trust formation is less of a technical challenge, but rather a complex social problem: There are hardly any technical means to gain trust in a perfect stranger! Before we gain confidence in an individual, we tend to challenge the person's trustworthiness iteratively in small increments. Users are likely to adopt similar attitudes towards unfamiliar systems.

2.2 Design patterns for authentication, authorization, access control

The patterns below aim at providing simple yet effective means for user authentication and authorization, and at preventing the system from accidentally locking out legitimate, benevolent users.

2.2.1 Pattern »Callback Authentication« (AAA_CBA)

On receiving a message or a connection request, it is often essential for an Aml system to verify the identity of the communication partner in order to prevent Denial of Service (DoS) or spoofing attacks, integrity violations, or the disclosure of private information. Unfortunately, it is not always possible or desirable to pre-arrange trusted credentials for a sound cryptographic authentication procedure. The AAA_CBA pattern (see p. 7) provides implicit authentication of a communication partner without the need to exchange credentials. It is based on the assumption that often the origin of a communication request provides sufficient proof of authenticity, and that it is difficult for an adversary to impersonate a legitimate communication endpoint.

Consequences

- *accountability*: Callbacks authenticate a location, not a user. Thus, a malicious user can more easily disclaim accountability by pretending that a different user spoofed his identity.
- *accountability*: An adversary may spoof the identity of a third party to make the system callback this claimed identity. From the third party's perspective, this could be interpreted as a DoS attack originating from the Aml system because the callback originates from the system, not from the adversary.
- *performance*: Callbacks cause a slight delay in session establishment, and they are only appropriate for session-oriented interactions. Connectionless communication (»datagram« transmission) is probably not a good candidate for this pattern.
- *cost*: If communication uses a public network and the user is billed for each connection, callbacks lead to reverse charging, imposing all tolls on the system.

Variants

Callbacks may be combined with public key cryptography as follows: The system uses a callback to transmit its public key back to the originator of the connection

Callback Authentication AAA_CBA This pattern provides implicit authentication of a communication partner without the need to exchange credentials. It is based on the assumption that the origin of a communication request provides sufficient proof of authenticity, and that it is difficult for an adversary to impersonate a legitimate communication endpoint.	
Background	On receiving a message or a connection request, it is often essential for an Aml system to verify the identity of the communication partner in order to prevent spoofing attacks, integrity violations, or the disclosure of private information. Unfortunately, it is not always possible or desirable to pre-arrange trusted credentials for a sound cryptographic authentication procedure.
Problem	How can we authenticate communication partners in an ad-hoc manner, without the need for pre-shared keys, public key infrastructures or lengthy authentication procedures?
Solution	To verify that a communication request originates from the claimed communication partner, we decline accepting the existing communication link, but initiate a new connection to the claimed origin on our own by doing a callback. Unless an adversary is able to successfully impersonate the callback destination, and to redirect our callback request to a spoofed communication endpoint, we can be reasonably sure that we are actually connected with the claimed location.
Issues	<p>Callback provides reliable endpoint verification only if the underlying communication technology prevents easy redirection of outgoing call attempts. Point-to-point connections or switched networks (e.g., PSTN) typically provide this characteristic. Routed networks (e.g., the Internet) are more susceptible of call redirections that pass unnoticed.</p> <p>Note that a callback only authenticates a communication endpoint, not the individual (or system) listening at the endpoint. Therefore, callback authentication is only useful if the endpoint enforces secure physical access: »Whoever listens at the endpoint is a legitimate communication partner.«</p>
Examples	<p>The classic example of a callback authentication occurs in the public switched telephone network, where we can assume that it is reasonably difficult to impersonate a known telephone number. For example, remote access to an intranet via modem is often secured by calling back the claimed originator of a connection request. If only known legitimate numbers are called back, an adversary may claim a wrong identity by signalling a spoofed calling line indication, but will fail to receive the callback.</p> <p>In wireless communication in the context of assisted living, a callback with low signal strength may be used to exclude impostors who try to intrude the internal communication of smart home appliances and in-door sensor networks by inserting system messages from the distance with a strong signal. We can prevent such intrusions as long as the adversary is unable to receive our weak callback signals from outside the household.</p> <p>The BelAml demonstrator may use callbacks to authenticate a medical care center, for example, the center's request for a telediagnosics session. A medical care center, in return, may use a simple callback to verify incoming emergency calls.</p>
Related Patterns	AAA_LDA »Location-dependent Access«

request. The originator can be reasonably sure that the key returned is in fact owned by the system. Subsequent communication can be encrypted with a shared key exchanged by the communication partners using this public key. Encryption adds confidentiality to the authenticated connection.

2.2.2 Pattern »Location-dependent Access« (AAA_LDA)

Access control requires proper authentication, which typically verifies the user's claimed identity based on something that he *knows* (e.g., a password), something that he *has* (e.g., a smartcard), or something that he *is* (a biometric property such as a fingerprint). Unfortunately, these authentication means are often too intrusive or impractical in the context of ambient systems. The AAA_LDA pattern (see p. 9) proposes to use the physical location of a user to discriminate legitimate from illegitimate system access.

Synonyms

This pattern is also known as »Spatial Access Control«.

Consequences

- *availability*: The system might lock out legitimate users whose current position cannot be located with sufficient reliability.
- *integrity*: Once a mobile user has gained access he might keep his connection alive while changing his location, thus invalidating the access prerequisite. Depending on circumstances, this might constitute a policy breach.
- *location privacy*: The pattern forces the user to disclose his current position just to gain access.

Variants

A simple variant of this pattern is »locality«: Collected data is not disseminated indefinitely, but strictly tied to the place at which it is collected. For example, if a sensor network identifies a user, this information is only made available within the same building, or within some pre-defined geographic boundary.

An interesting solution to the »notification & consent« principle of the Fair Information Practices is the »proximity principle«: Data collected by a smart environment is made available to a user if and only if he witnessed the situation in which it has been collected. For example, a video recording of a meeting is

Location-dependent Access AAA_LDA	
The physical location of a user is a useful attribute to discriminate legitimate from illegitimate system access.	
Background	Access control requires proper authentication, which typically verifies the users claimed identity based on something that he <i>knows</i> (e.g., a password), something that he <i>has</i> (e.g., a smartcard), or something that he <i>is</i> (a biometric property, e.g., a fingerprint). Unfortunately, these authentication means are often too intrusive or impractical in the context of ambient systems.
Problem	Is there a simple, unintrusive way to restrict access to »insiders«?
Solution	The metaphor of an »insider« — someone who is legitimately involved in a process or enterprise — can be taken literally to indicate a potential solution: Being »inside« of an Aml-enabled household or being in the immediate vicinity of a service provision point may be sufficient justification to provide access. Location-dependent access control can be based either on <i>absolute</i> user location or on the users' <i>relative</i> locations.
Issues	Location-dependent access control requires that <ul style="list-style-type: none"> • either the identity of the user is irrelevant, provided the user is at the claimed location, or • physical access control can enforce that only legitimate users can access the location. In both cases, the AAA_LDA depends on a sufficiently strong mechanism to verify the claimed location of the user. Thus, the pattern must be used with care if location tracking uses wireless technologies or unsecured communication links.
Examples	A typical access control policy based on relative locations could be: »My colleagues may track my location provided that we are in the same building«. As another example, access to protected health information in an assisted living scenario may be restricted to caregivers inside the assisted person's household; to verify her location, and to gain access to the local Aml system for a pre-defined time span, the caregiver may be required to synchronize her Digital Nurse Assistant with a near-field communication interface inside the clients home. It is good practice to restrict administrative (»superuser«) system access to dedicated admin terminals, and to physically restrict access to these terminals. In fact, the »Callback Authentication« (AAA_CBA) pattern is another example of location-dependent access: Whether the originator of a communication is granted access to a communication channel is determined solely on the basis of his current location — his presence at the claimed (and trusted) communication endpoint.
Related Patterns	AAA_CBA »Callback Authentication«

accessible only to the participants of that meeting. The key idea is that meeting participants can more light-heartedly agree to such an access policy because access is only granted to those who know anyway (but may have difficulties to recall exactly) what happened during the meeting. Agreeing to a general »proximity policy« would spare the user repeatedly expressing explicit consent to data collection.

Instead of location-dependent restrictions the designer may consider temporal restrictions to control access to personal data. For example, location tracking may be enabled »only during business hours« (i.e., while the user is presumably at work). Or, access to a patient's medical record may be provided »only while at least one doctor is logged into the system« (so that she can see a privacy alert on her desktop that allows immediate prosecution of unauthorized access).

References

In [3], Bullock and Benford propose SPACE, an access control framework for multi-user collaborative environments. It provides unintrusive access control to data spatially divided among different (virtual) regions. With wireless authentication, an ambient system may employ similar concepts to provide location-dependent access control

In [35] Tolone et al. survey different approaches for access control in the context of collaborative systems and discuss their relative merits and shortcomings.

2.2.3 Pattern »Emergency Access« (AAA_EMA)

A system may deny access to users for several mundane reasons, not necessarily implying a security threat: The user may have simply forgotten his password. In an emergency situation, denial of access may cause a serious crisis if a competent potential user is available but lacks formal accreditation to the system.

Therefore, some security standards, such as the regulations of the U.S. Health Insurance Portability and Accountability Act (HIPAA) [9], strictly require emergency access procedures to prevent inappropriate service degradation due to system unavailability. Sometimes, access should be granted although the user is unable to provide proper credentials. The AAA_EMA pattern (see p. 11) suggests to compensate insufficient possibilities to authenticate the user by restricting user access with suitable guard conditions.

Emergency Access AAA_EMA	
Sometimes, access should be granted although the user is unable to provide proper credentials. This pattern compensates for insufficient possibilities to authenticate the user by restricting user access with suitable guard conditions.	
Background	A system may deny access to users for several reasons: The user may have forgotten his password or lost his cryptographic token, or the iris scanner may be misaligned causing repeated false negatives. Or, a competent potential user is available in an emergency situation but lacks formal accreditation to the system. Therefore, some security standards such as the regulations of the U.S. Health Insurance Portability and Accountability Act (HIPAA) [9] strictly require emergency access procedures to prevent inappropriate treatment due to system unavailability.
Problem	How can we provide legitimate system access for all contingencies?
Solution	Access control mechanisms (and regulations!) should provide exceptions for emergency situations. That is, it should be possible to circumvent normal access control under well-defined guard conditions, and maybe subject to certain restrictions. For example, a user may disable access control altogether by pushing an »emergency button«, provided that pressing the button will cause an alert notification to the system operator, and maybe video surveillance of the emergency terminal. Furthermore, emergency access mode may be limited to a restricted time span and to a few essential functions, and each access should be logged meticulously. And maybe the emergency button is disabled as long as a legitimate user is logged into the system.
Issues	To compensate for the lack of access control, suitable guard conditions must be defined and enforced. For example, to hold a user liable for an unjustified emergency access this user must be reliably authenticated, or he must be »arrested« until he can be held responsible. Enforcement of guards can be beyond the scope of systems design, implemented, for example, with procedural safeguards as part of a comprehensive contingency plan.
Examples	In an assisted living scenario, a doctor, nurse, or paramedic arriving at the scene after an emergency call can typically not wait until an accredited caregiver is available to unlock the medical records of the local Aml database. If an emergency notification arrives at the medical care center, the help desk may want to start video surveillance for a closer assessment of the situation; however, if the assisted person is actually injured, she may be unable to positively acknowledge a video connection, which would violate privacy rights under normal conditions. In both scenarios, we could simply overrule access control by pushing an emergency button, justifying our decision later, based on an audit trail of our system access.
Related Patterns	

Synonyms

This pattern is referred to as »Guarded Access« in [5]; note, however, that the same terminology is used for unrelated architectural patterns, too.

Runtime dynamics

Monitoring of emergency access facilities is essential to prevent abuse. Furthermore, procedures must be in place for »cleaning up« after an emergency access, including these tasks:

- disabling of emergency access (e.g., invalidating existing emergency passwords) and restoring normal access control,
- evaluating the relevant audit trails of emergency access facilities,
- preparing for the next emergency (e.g., assigning and distributing new emergency passwords),
- reviewing the course of the emergency to determine efficiency and effectiveness of emergency procedures, and to adjust if necessary.

Although most of these tasks are procedural safeguards, they may be supported by technical controls. For example, the system can secure audit trails automatically, and it may autonomously disable emergency facilities after a predefined maximum emergency period.

Consequences

- *accountability*: An adversary may use emergency access facilities for a very quick access, vanishing before their use is recognized (similar to setting off a false fire alarm by smashing the glass of an alarm button). Safeguards must be in place to prevent long delays between the first activation of emergency access facilities, and the starting of their proper surveillance and control.
- *privacy*: Users should be aware that emergency access is subject to extensive monitoring. Under emergency conditions, privacy of system use may be severely restricted.
- *usability*: To limit abuse potential, only a subset of functionality might be available for emergency access. There is a conflict between maximum security and maximum effectiveness for quick emergency response.

References

The Joint NEMA/COCIR/JIRA Security and Privacy Committee (SPC) published an approach to granting emergency access to medical systems [34]. In this white paper, the SPC stresses the need for pre-staged emergency accounts for medical information systems, which is mandated by U.S. and European legislation.

2.2.4 Pattern »Slave Imprinting« (AAA_SLI)

In an Aml environment with many devices that need to cooperate in an ad-hoc style in changing configurations, there is a need for transient, revocable associations between devices. The AAA_SLI pattern (see p. 14) is a means to establish such associations securely, and with minimal resource requirements.

Synonyms

This security pattern is better known under the name of »The Resurrecting Duckling«.

Consequences

- *availability*: If the master (and its imprinting key) is lost, we might be unable to revoke imprinting (i.e., to »assassinate« the duckling) in order to imprint it to a new master. Depending on circumstances, an escrow key should be provided to support instant replacement of a master device.
- *integrity*: In hostile environments physical integrity is important to prevent an adversary to impersonate the slave, or to manipulate the slave's behavior; furthermore, the slave device requires protection against unauthorized software upload. In a more secure environment such as the private household of the user, simple »lock/unlock« and »reset« buttons on the slave device may suffice for reasonably secure imprinting and re-imprinting.
- *confidentiality*: Imprinting the slave device establishes a shared key that can be used for subsequent encryption of communication. Initial key exchange could be based on a secure channel such as a wired connection, or a (shielded) near-field communication link.

References

The pattern and its catchy name »Resurrecting Duckling« were coined by Stajano and Anderson [30], based on an analogy taken from biology, namely the imprinting process between a mother duck (master) and the duckling (slave). In

Slave Imprinting AAA_SLI The pattern provides secure, transient, and revocable association between a master and a slave device.	
Background	In the ubiquitous computing world, you no longer want to litter your coffee table with an array of remote controls for your TV, stereo, DVD, VCR, curtains, central heating, and air conditioning. Instead, you want all of these systems to obey a universal remote control. Because you no longer buy the remote control with the appliance, you need to be able to establish an association between the two after purchasing the appliance. Because you don't want your neighbor to be able to activate your appliances (whether by accident or malice), you want this association to be secure. And, because you want to be able to resell your old stereo while keeping your remote control, and you want to be able to replace a broken remote control without losing control of all your appliances, you also want this association to be transient, or revocable.
Problem	How can we establish a secure, transient association between devices that provide only scarce resources?
Solution	Four principles define this pattern: <ul style="list-style-type: none"> • <i>Two states</i>. The slave can be in one of two states: imprintable or imprinted. In the imprintable state, anyone can take it over. In the imprinted state, it obeys only its master. • <i>Imprinting</i>. The transition from imprintable to imprinted happens when the master sends an imprinting key to the slave. This must be done using a channel whose confidentiality and integrity are adequately protected. • <i>Death</i>. The transition back from imprinted to imprintable is known as death, and can only be initiated by an order from the master, or spontaneously according to a predefined policy. (E.g., the master-slave association resolves automatically after a given time period or after each successful completion of a single transaction.) • <i>Assassination</i>. The slave must be constructed in such a way that it will be uneconomical for an attacker to assassinate it, that is, to cause the slave death artificially in circumstances other than the one prescribed by the death principle.
Issues	To prevent impersonation or manipulation of an imprinted slave in hostile environments, the slave device should be tamper-proof, at least it should provide tamper-evidentness (i.e., manipulation should not be possible without breaking a seal).
Examples	One application is the secure integration of new domestic appliances into the local control environment. A household should provide sufficient physical protection against malicious access so that tamper-resistance is probably not an issue for typical consumer durables. Another scenario could be a short-term spontaneous association between a personal device and some peripheral selected from a pool of available peripheral devices. For example, in a hospital there may be a bowl of disinfectant containing ten thermometers. The doctor does not really care which thermometer she gets when she picks one up, but she does care that the one her palmtop talks to is the same one she is holding and not any other one in the bowl or nearby in the ward. Imprinting could also be useful to establish a secure connection between an electronic wallet and an automated teller machine for a single transaction.
Related Patterns	

a follow-up paper [31] Stajano extended the pattern from hierarchical master-slave relations to more symmetric peer-to-peer scenarios.

2.2.5 Pattern »Plutocratic Access Control« (AAA_PAC)

Small or wireless devices might easily fall victim to resource depletion, especially to DoS attacks aiming at battery exhaustion. The AAA_PAC pattern (see p. 16) tries to prevent malevolent peers from indiscriminately asking for resources.

Consequences

- *accountability, availability*: In an Aml setting where users are typically not charged monetary values but rather some other kinds of sacrifices, the pattern can hardly enforce strict access control. The need to balance security and availability often necessitates rather liberal charging policies. The designer must be careful not to discriminate legitimate users simply because they are not able to pay in the required »currency«.
- *usability*: »Plutocratic Access Control« can deter a rational adversary who carefully weighs costs against benefits; it cannot prevent irrational (or blind) DoS attacks.
- *performance, cost*: Requiring some sort of »payment« for each request will most likely slow down the service.

Variants

The client or peer might be charged in all kinds of »currencies«. For example, it may cost significant computing power to formulate a request. Or the server may simply delay his answer to force the client to spend some waiting time. Alternatively, it might be intellectually challenging to receive an answer, for example, because the server counters with a puzzle before answering a user's request to differentiate between human and machine users.

References

The term »Plutocratic Access Control« was coined by Stajano and Anderson, and the idea appeared in a revised version of [30] that appeared in IEEE Computer, Vol. 35(4), in April 2002² as well as in Stajano's textbook [32].

² see <http://csdl.computer.org/comp/mags/co/2002/04/r4s22.pdf> for an electronic version

Plutocratic Access ControlAAA_PAC The pattern is one approach to defend a system against Denial-of-Service attacks.	
Background	Even with strong cryptography in place for proper authentication, a device has difficulties to repel DoS attacks: Before a request can be rejected as impermissible the device has to put some effort in analyzing and authenticating the message. Mobile devices with scarce resources are therefore vulnerable by the so-called »sleep deprivation torture«.
Problem	How can we defeat Denial-of-Service attacks with limited energy resources?
Solution	<p>An approach to this problem is plutocratic access control: you receive service only if you pay for it. By charging for access, the server limits the extent to which clients can indiscriminately ask for resources. In fact, if the charge is such that the server makes a profit in serving a user, the DoS problem may no longer be a concern: exhaustion of the available capacity simply means that the server has made as much money as it possibly could!</p> <p>The pattern is not restricted to charging actual money. The server can use the same limiting strategy by forcing users to undergo some expensive sacrificial ritual in exchange for service. For example, servers could make clients solve cryptographic puzzles or answer a question that would be easy for a human but hard for a machine. The latter might be more suited to peer-to-peer applications, while the former might be better in ubiquitous computing environments.</p>
Issues	»Plutocratic Access Control« can deter a rational adversary who carefully weighs costs against benefits; it cannot prevent irrational (or blind) DoS attacks.
Examples	The »exponential backoff« login strategy increases the delay between successive login attempts to prevent password-guessing by exhaustive search. In this example, the waiting period is the charge a client must pay for each login.
Related Patterns	

3 Privacy and Anonymity

Privacy is one of the most often cited concerns of modern computing. The Aml paradigm exacerbates the privacy problem [15], because Aml systems tend to collect sensitive and personal data such as an individual's location and activity; sometimes they do so unnoticed by the individual and for unpredictable reasons — »just in case the data may turn out to be helpful in the future«. Another problematic feature of Aml systems is memory amplification [17]: Not the least bit of a user's embarrassing activities is ever mercifully forgotten.

Many complex privacy issues need to be handled, such as what information can be collected, how and from whom. Privacy laws vary from country to country, and from domain to domain (e.g., health care, telecommunication, electronic commerce), but they more or less agree on a common set of core principles.

3.1 Peculiarities of privacy

Privacy has some characteristics that are difficult to enforce in an Aml environment.

Enforceability, detectability and recoverability

IT security comprises attributes such as integrity, availability, and confidentiality; Roughly speaking, the latter is almost a synonym for privacy — at least privacy and confidentiality imply similar protection measures. All these security attributes are so-called non-functional qualities, but there is a remarkable difference between availability and integrity on the one hand, and confidentiality / privacy on the other:

- Integrity and availability are (ultimately) *not enforceable*, but *breaches* are *detectable* and *recoverable* with redundancy techniques.
- Confidentiality and privacy are *enforceable* with cryptology, but *breaches* are *neither detectable nor recoverable*.

That is, in general we cannot recognize when privacy in a system is lost because eavesdropping does not affect the system state. Even worse, once privacy has been compromised, the damage cannot be undone because our confidential

information is no longer a secret. These two characteristics make privacy problems particularly difficult (and expensive) to solve: Our only line of defense is proactive privacy enforcement — there is no adequate compensation for privacy breaches after the fact.

User consent

Privacy is not an absolute quality. Most of the time, there is a trade-off between privacy and service level. The user can trade loss in privacy for better quality of service, but the system does not know the user's relative appreciation of privacy versus service quality. Thus, before the system triggers an action that may harm the user's privacy, explicit user consent is required to ensure that from the user's perspective the gain in service quality outweighs the implied loss in privacy.

However, true consent requires true choice. But a user can only reasonably choose

- if he has all relevant information that affects the choice,
- if he is able to understand all implications of his final decision,
- and if an opt-out decision does lead to undue discrimination («digital divide«).

Although these preconditions are plain obvious, they are hard to satisfy in practice — in particular in an Aml environment. For example, how could we recognize even *the need* to express our consent to sensor surveillance that has been designed to be »invisible«? How can we be sure that the system will respect our preferences if the sensor network can operate »behind our backs« without any user cooperation? How can we express privacy preferences or consent in a system without explicit user interfaces? How can the user acknowledge the accuracy and completeness of complex data without prior in-depth analysis of all implications, interrelations, and potential future uses of the data. And can we reasonably expect that the privacy preferences of all participants attending a meeting can be reconciled without sacrificing even the most innocuous sensor capabilities of the system?

These examples show that the legal aspects of privacy are probably as intricate as its technical aspects.

3.2 General privacy principles

The so-called »Fair Information Practices« were first formulated by the U.S. Department of Health, Education, and Welfare in 1973. They are the basis for

many privacy laws and guidelines today, such as the recommendations of the Organization for Economic Cooperation and Development (OECD). In essence, the OECD's Guidelines on the Protection of Privacy and Transborder Flows of Personal Data [21] require

- *Respect for user privacy*: »Everything is private by default!«
- *Transparency*: »Who collects what and when, and for what purposes?«
- *User notification*: »The user receives adequate feedback on the collection of data flows (the reference data), traffic data (analysis of communication relations), and location data (current user position and movement trace).«
- *User consent*: »The user is explicitly asked to agree to the collection of data flows, traffic data, or location data, and to the disclosure of such data to a third party.«
- *Opportunity for opt-out*: »The user is free to decline data collection requests without suffering discrimination.«
- *Data parsimony and legitimation*: »The *need to know principle* is applied with respect to volume of data collected, retention period, and subgroup of people to whom the data may be disclosed.«
- *Purpose binding*: »Private data must only be collected for a well-defined purpose, and this purpose must be defined in advance of collection.«
- *Proportionality*: »Privacy intrusion must be in proportion to the purpose at hand (i.e., the data collected must be relevant and adequate for the intended use).«
- *Accuracy, completeness, and validity*: »Collected data must be reliable and up-to-date, and it must convey the full, undistorted picture with respect to the intended use.«
- *Opportunity to challenge validity*: »The user can inspect the collected data, and the collector has the obligation to correct invalid data on request.«
- *Information shelter*: »The user has an opportunity to withdraw from data collection (i.e., a zone of unobservability); the user has a right to refuse information reception at any time (e.g., to prevent spam or sensory overload).«
- *Security safeguards*: »Personal data should be protected by reasonable security safeguards against such risks as loss or unauthorized access, destruction, use, modification or disclosure of data.«

Furthermore, the data collector is held accountable for complying with these measures. The European Community (EC) codified these fundamental privacy principles for data processing and communication in the directives 95/46/EC and 2002/58/EC [6, 7].

3.3 Design patterns promoting privacy

Privacy enhancement is better obtained by actively constructing a system exactly tailored to specific goals than by trying to defend ex-post a poor design against misuse or attacks.

The subsequent patterns were mostly inspired by a collection of more general patterns for ubiquitous computing, compiled by the *Group for User Interface Research* (GUIR) at the University of California at Berkeley [5]. Other useful guidance on Aml privacy strategies and principles may be found in [14, 16].

3.3.1 Pattern »Privacy Sensitive Architecture« (PRI_PSA)

The PRI_PSA pattern (see p. 21) serves as a reminder for some fundamental design decisions that promote privacy in line with Fair Information Practices. These design decisions have to be taken early because they deeply affect the structure and dynamics of the system.

Synonyms

This pattern is also known as »Privacy-aware System« (PawS).

Consequences

This pattern has far-reaching implications as it postulates the application of a number of more elementary sub-patterns.

- *usability*: Privacy and usability are often antagonists, because the obligation for data parsimony, user notification, and user consent may interfere with a smooth workflow.
- *performance, cost*: Comprehensive support for privacy is likely to increase the complexity of the required equipment, functionality, and data. This will adversely affect performance and cost of the system.

Privacy Sensitive Architecture PRI_PSA This pattern serves as a reminder for some fundamental design decisions that promote privacy in line with Fair Information Practices. These design decisions have to be taken early because they deeply affect the structure and dynamics of the system.	
Background	Privacy cannot be bolted on an existing Aml system subsequently, but requires fundamental architectural support to be effective. A privacy-sensitive architecture is also important to achieve user acceptance: explicit architectural support for privacy can influence how the system is perceived and used.
Problem	How can we provide structural support for privacy protection? How can we demonstrate to the user that we address privacy concerns adequately?
Solution	From the outset, we have to plan for the fundamental privacy principles, in particular data parsimony and proportionality, transparency, and information shelter. Building blocks for a solution are: <ul style="list-style-type: none"> • Provide physical privacy zones that are strictly excluded from video, audio, and sensor surveillance; for each zone that is not fully privacy-safe, provide a clear policy that defines what personal data is being captured. • Avoid the need for users to actively emit signals. Rather, use an »information beacon« paradigm where the user passively listens to information broadcasts; let the user decide whether he wants to reveal private data, and to what extent (push model as opposed to the system pulling data outside the user's control). • Try to concentrate sensitive information and processing in a trusted device, ideally a Personal Digital Assistant (PDA) under exclusive user control. • Provide data (e.g., sensor readings, video, identity) with the minimum accuracy and resolution required for the intended purpose; deliberately blur the accuracy by spatial or temporal aggregation. • Provide comprehensive but unintrusive feedback about privacy-critical system activities (i.e.: What data is tracked, stored, or transferred to a third party? How, when, and why?).
Issues	The user is generally unable to tell whether the system actually follows the ostensive privacy-sensitive architecture, or whether it recognizes Fair Information Practices only superficially, that is: cheats! Trust must be provided by non-technical means, for example, legal obligations or reputation. Fair Information Practices are inconsistent with the Aml paradigm of unintrusiveness (»invisibility«) and ad-hoc support (»collecting sensor data for an unforeseen demand that may arise in the future«). It is an open question whether these conflicting goals can be reconciled, or whether the Aml vision requires some amendments.
Examples	Instead of active location tracking, we may give preference to a design where the user determines his location by passively listening to beacons that broadcast positioning information; the user decides himself when and to whom he reveals location information. We may put the user in control by keeping all personal data in a trusted, personal device, answering information requests from the system about identity, location, credentials, or preferences only selectively, and with deliberately reduced accuracy.
Related Patterns	PRI_PPZ »Physical Privacy Zone« PRI_INV »Invisibility Mode« PRI_BCN »Beacon« PRI_TPD »Trusted Personal Device« PRI_BPD »Blurred Personal Data« PRI_PFB »Privacy Feedback«

References

The compilation of ubicomp patterns available at the GUIR home page (see web link [5]) provides further discussion on the PRI_PSA pattern and related patterns.

3.3.2 Pattern »Privacy Enhancing Technology« (PRI_PET)

A system design should strive for anonymity of system users. Avoiding collection and storage of identifiable data is preferable to protecting it. If data identification cannot be avoided, the system should use unlinkable pseudonyms that are only disclosed on a need-to-know basis. The PRI_PET pattern (see p. 23) aims at eliminating or minimizing identifiable data thereby preventing unnecessary or unwanted processing of personal data, without the loss of the functionality of the information system.

Consequences

- *privacy*: Data mining techniques may be used to recombine allegedly anonymous pieces of information to identifiable data records. It is quite difficult to prevent circumvention of PET measures by a »subliminal channel« in the physical world outside the IT system. For example, simple coincidence of two real-world events may provide the vital clue to the observer that the anonymous data involved in these events is actually related.
- *privacy*: Concealing identities in stored or transmitted data may be insufficient to achieve anonymity, because the origin and destination of communication often reveals identification information. Likewise, it is not necessary to know *what* is being communicated as long as we can apply traffic analysis to determine *how much* data is being sent, *and when* it is being sent. These observable parameters often provide sufficient clues to disclose the identities of the communication partners. To prevent traffic analysis and message traceability, so-called MIX networks may ultimately be required — a mechanism that is probably too resource-consuming for typical Aml environments.³
- *accountability*: If we cannot prevent collection and access to identifiable data, we must carefully log the access operations to hold the user accountable for any illegitimate use of private information.

³ Note also that the principle of obfuscation on which a MIX network is based essentially requires routed, multi-hop transmission. It is undermined in wireless single-hop networks where all participants may directly observe the sending of the original message.

Privacy Enhancing Technology PRI_PET	
This pattern aims at eliminating or minimizing personal data thereby preventing unnecessary or unwanted processing of personal data, without the loss of the functionality of the information system.	
Background	The principle of »data parsimony« requires that only a minimal amount of personal information is collected, and that it should be stored only for a minimal retention period. Avoiding collection and storage of identifiable data is preferable to protecting it.
Problem	How can we reduce the processing of privacy-sensitive data to a minimum without significant loss of Aml functionality?
Solution	<p>»Privacy Enhancing Technology« stands for a coherent system of measures that protects privacy by eliminating or reducing personal data or by preventing unnecessary and/or undesired processing of personal data. The building blocks of PRI_PET are:</p> <ul style="list-style-type: none"> • Apply the »privacy razor« [14] to eliminate any parameter and functionality carrying personal data that is not strictly necessary. Can we provide equivalent functionality without using identity or other person-related information. • Divide the system into identity, pseudo-identity and anonymity domains, and separate data and functions accordingly. • Employ an <i>identity protector</i> component (device or organizational unit) to remove identifiers (»de-identification«, anonymization) or replace them by pseudonyms, and use the identity protector as a guard for data transfers between identity, pseudo-identity and anonymity domains. • To link de-identified data to an identity, use pseudo-identities (»pseudonyms«), for each pseudo-identity domain a different one. Make sure that different pseudo-identities referring to the same identity are mutually unlinkable. • If person-related information cannot be removed, encrypt it. • Apply strict access control on a need-to-know principle. As a prerequisite, enforce secure authentication, and assign roles with suitable access profiles to each authenticated individual. • Make sure that data access is properly and securely logged in an audit trail. For the audit trail to be meaningful, the system has to enforce reliable authentication.
Issues	Data mining techniques may be used to recombine allegedly anonymous pieces of information to identifiable data records. Knowledge about the physical world (i.e., information that is not necessarily available in electronic format) may be used to bridge the gap between linkable but pseudonymous electronic data and the user's identity. For example, if a patient with pseudonym X is charged for staying in single room 100 in hospital, and an observer saw Joe Smith entering and leaving this room during the period of X's staying time, then the observer might conclude that the true identity of X is probably Joe Smith.
Examples	There is no need to know the full postal code of the data subject for statistical purposes. It suffices to know the region where the data subject lives. The full postal code must, therefore, be truncated to the required length.
Related Patterns	PRI_PSA »Privacy Sensitive Architecture« PRI_PAI »Partial Identification«

- *integrity*: »Privacy Enhancing Technology« (PET) must be implemented robustly so that an adversary cannot circumvent or disable its protection mechanisms. On the other hand, de-identification of personal data must take care not to change the meaning of the original data.
- *performance, cost*: Although PET entails some up-front costs in terms of money and performance, reducing the amount of sensitive data that has to be managed and protected should save costs in the long run.

References

A detailed discussion of privacy enhancing technologies can be found in the *Handbook of Privacy and Privacy Enhancing Technologies*, published by the PISA Consortium [2] in 2003.

3.3.3 Pattern »Blurred Personal Data« (PRI_BPD)

Video cameras, microphones, or sensor networks often deliver more private information than is strictly necessary for the intended use of the system. The principles of data parsimony and data proportionality mandate that a minimum amount of private information should be collected, stored, and transmitted. Deliberately removing details from available data offers fine-grained control of the amount of information we want to disclose, and it reduces the damage caused by unauthorized disclosure. The PRI_BPD pattern (see p. 25) suggests different approaches for blurring personal data.

Consequences

- *accountability*: Reduced fidelity of collected data may interfere with proper user authentication and meticulous logging.
- *accountability, usability*: In some situations it might be embarrassing for the user if other users can find out that transmitted data has been blurred deliberately. The designer must consider this possibility; in critical situations blurring either should be indiscernible for other users, or it should be indistinguishable from limited sensor accuracy.
- *usability, performance*: There is often a fidelity trade-off between the amount and accuracy of disclosed data and the quality of service provided by the system: Less accuracy may lead to poorer service. These trade-offs may be difficult to communicate, preventing the user from taking a reasonable choice.

Blurred Personal Data PRI_BPD The principles of data parsimony and data proportionality mandate that a minimum amount of private information should be collected, stored, and transmitted. By deliberately removing details from available data, we can control how much information we want to disclose.	
Background	Video cameras, microphones, or sensor networks often deliver more private information than is strictly necessary for the intended use of the system. This leaves room for improved data parsimony.
Problem	How can we minimize privacy disclosure, and how can the user gain better control about the degree to which he disclosed his privacy?
Solution	<p>In many situations the system requires only approximate data to satisfy the user's needs. Thus, if queried by the system the user is free to provide information at coarser granularity. If the system tracks private user information autonomously, the Fair Information Practices suggest that it should do so at the least tolerable level of accuracy.</p> <p>There are several approaches to decrease the precision of available information:</p> <ul style="list-style-type: none"> • <i>aggregation</i>: several individual data records can be combined into a single compound records (e.g.: »twice last week« instead of »on Tuesday and Friday«, »five on average« instead of »three, five, and seven«) • <i>reduced resolution</i>: (e.g.: »in Kaiserslautern« instead of »Fraunhofer-Platz 1«, 20 pixels per inch instead of 500 pixels per inch). • <i>obfuscation</i>: adding random noise to the signal before disclosure • <i>k-anonymity</i>: concealing the true data in a set of k possible values (e.g.: »I am currently in Frankfurt, Kaiserslautern, Saarbrücken, or Karlsruhe; send me a city map«) • <i>reduced freshness</i>: providing accurate data, but only about some status in the past (e.g.: »You can track my location info if it is older than a day; but I will not disclose my current location.«).
Issues	For data actively disclosed by the user (e.g., the user's PDA answering a request from the surrounding Aml system) blurring is at the user's sole discretion. However, if the data is autonomously tracked by the system (e.g., location tracking, video surveillance) then the user has no choice but to trust the system. There is often a fidelity trade-off between the amount and accuracy of disclosed data and the quality of service provided by the system: Less accuracy may lead to poorer service. These trade-offs may be difficult to communicate, preventing the user from taking a reasonable choice.
Examples	To determine the occupancy of a meeting room, high-definition video surveillance is not required. Low-resolution images of a few pixels per inch should suffice to detect motion, to concentrate the lighting to the position of the occupants, and to distinguish between humans and vacuum cleaners. In this scenario, low resolution prevents the identification of individual persons.
Related Patterns	

- *cost*: On the one hand, reducing the volume and fidelity of collected data are likely to save costs; techniques such as obfuscation or even k-anonymity (see p. 25 for an explanation), on the other hand, cause additional overhead and consume some bandwidth.

References

The compilation of ubicomp patterns available at the GUIR home page (see web link [5]) provides illustrations of blurring techniques and further references.

3.3.4 Pattern »Partial Identification« (PRI_PA)

The user's identity has many facets that can be characterized by different attributes. For many intended system uses, only a few attributes of the user's identity must be revealed to provide adequate service and security. Rather than requiring precise identity, some systems could just confine themselves to a selected subset of identity attributes, a so-called partial identity profile. The PRI_PA pattern (see p. 27) looks at different aspects of identity.

Consequences

- *accountability*: Reduced accuracy in user identification constrains prosecution of the adversary if the system suffers an attack.
- *usability*: there may be a privacy/quality trade-off: The fewer information the user provides, the fewer service he will receive. For example, inaccurate identification conflicts with user-aware service tailoring.

References

In his master thesis [19] Nguyễn presents a framework to formalize the notion of identification systems, and he gives an overview of the concepts and definitions that are involved — including partial identification.

3.3.5 Pattern »Beacon« (PRI_BCN)

An Aml system that requires active client participation to provide service forces the user to reveal his presence. Furthermore, any RF signal emitted by a personal device may be »RF fingerprinted« for subsequent re-identification, even if the transmitted data does not contain any explicit ID tag or personal information that can be traced back to an individual. To better promote location privacy, the PRI_BCN pattern (see p. 29) proposes an architecture based on information

Partial Identification PRI_PAI For many intended system uses, only a selected subset of the user's identity must be revealed to provide adequate service and security.	
Background	The user's identity has many facets that can be characterized by different attributes. Rather than requiring precise identity, some systems could just confine themselves to a selected subset of identity attributes. This pattern looks at different aspects of identity.
Problem	People may not fully trust a system to handle their personal information properly. Can we build systems that do not require full identity to work properly. What alternatives are there to full identity, and how can they be applied?
Solution	<p>An alternative to requiring full identity is to use partial identity. The key here is separating identity from actual characteristics that are needed, and checking only the aspects of identity that are required. There are several approaches:</p> <ul style="list-style-type: none"> • classify but do not identify: e.g., identifying a »person« is enough to start lighting, ventilation and air conditioning in a conference room. • concentrate on selected characteristics: for example, being »an adult« (identified by a certain minimum weight and body height) might be sufficient to unlock the medicine cabinet in the bathroom. • check credentials that are independent of the user's identity: for example, »anyone with a smartcard loaded with enough electronic cash« is granted access — whoever it may be. <p>Ideally, different partial identities should be unlinkable, that is, it should be impossible to trace back the different identity attributes to a common individual.</p>
Issues	Partial identities based on selected characteristics are more easily spoofed than full identities. Furthermore, there may be a privacy/quality trade-off: The fewer information the user provides, the poorer service he will receive.
Examples	In [4] Carmenisch and Van Herreweghen describe the <i>idemix</i> anonymous credential system, which has the following properties: 1. Service providers know users only by pseudonyms; 2. Different pseudonyms of the same user cannot be linked. 3. An organization can issue a credential to a pseudonym, and the corresponding user can prove possession of this credential to another organization who knows him by another pseudonym without revealing more than the fact that the user owns such a credential. — These properties together enable protocols for fully anonymous, attribute-based, yet accountable subscription to a service.
Related Patterns	AAA_LDA »Location-dependent Access«

»beacons« provided by the system, to which the user can passively listen without a need to actively query the system.

Synonyms

In the literature, beacon-based positioning systems are sometimes referred to as »location-support systems« or »self-positioning systems« in contrast to »location-tracking systems« that actively locate a passive user or user device.

Consequences

- *integrity*: Leaving it to the user's discretion to respond — and to respond honestly and correctly — to a beacon might promote deception, jeopardizing the system's context awareness.
- *usability*: Responding to beacons should not require too much manual intervention. However, there is a conflict between level of automation and level of user control.
- *performance*: Beacons consume bandwidth even if their information is not required by any user. Thus, the beacon paradigm must be applied with care, and only for selected information assets.
- *cost*: A beacon-based architecture requires more sophisticated user equipment because it is the user's obligation to receive and process the beacon information. If a service forces the user to constantly listen to beacons this might rapidly deplete the energy resources of the user's personal device.

Variants

In general it is a good idea to let the user actively *push* personal data to the system on request, instead of letting the system automatically *pull* the data from the user's personal devices without explicit user participation (or under control of a user-defined privacy policy enforced by the user's personal device). Thus, selective answer under the user's control is not restricted to beacons.

An interesting idea is a »Privacy Beacon« [16]: The system uses the »Beacon« pattern to broadcast a machine-readable specification of its privacy policy. The user's »Trusted Personal Device« (see below) receives this specification and compares it with the user's privacy preferences, which are stored on the device (or on a privacy proxy reachable by the device) in machine-readable format, too. If policy and preferences match, the personal device starts cooperation with the

<p>Beacon PRI_BCN</p> <p>By emitting RF signals, a user reveals his presence to the Aml environment. To better promote location privacy, this pattern proposes an architecture based on information »beacons« provided by the system, to which the user can passively listen without a need to actively query the system.</p>	
Background	An Aml system that requires active client participation to provide service forces the user to reveal his presence. Furthermore, any RF signal emitted by a personal device may be »RF fingerprinted« for subsequent re-identification, even if the transmitted data does not contain any explicit ID tag or personal information that can be traced back to an individual.
Problem	How can we provide a privacy-friendly design that helps the casual user to conceal his presence and identity?
Solution	Instead of delivering service only in response to a user query, the system may proactively provide useful information that the user's personal device can silently utilize. Based on this information, the personal device can decide whether it is worthwhile to emit any RF signals to have access to enhanced service quality that requires active user participation. Depending on the user's preferences and available services offered, the personal device may decline to react to stimuli from the Aml environment.
Issues	<p>Many services require active client participation, so the user has no choice but to answer system queries if he wants to use the service. Even in this case, the beacon pattern may be used to inform the user about locally available services and privacy / service level trade-offs. At least the user has a chance for silent opt-out. Excessive use of the »Beacon« pattern consumes too much of the available bandwidth to be feasible; only a limited amount of data can be reasonably distributed with broadcasts.</p> <p>In a beacon-based architecture, the system must trust the information provided by its users; this is not an issue if only the user suffers a loss if poor (or plain wrong) data is provided.</p>
Examples	<p>Location tracking is a classical example where the user is forced to permanently emit »I am here« signals so that the system can localize him. A more privacy-sensitive solution is a self-positioning system based on beacons that constantly broadcast positioning information (»you are here«), which the user's personal device may passively receive and combine to obtain its current location. The Global Positioning System (GPS) is based on this principle. In a beacon-based positioning system the user knows where he is, while the system does not unless the user decides to reveal his presence: Even then, the user can disclose his position with only reduced accuracy. (Of course, a sophisticated system may try radio bearing to locate the RF sender, thus undermining the user's attempt to blur his location data.)</p> <p>Service advertising is another application of the »beacon pattern«. For instance, instead of having the user to query the environment for the nearest restaurant, the more private approach is to broadcast the positions of all restaurants within some diameter. The user may then compare these locations with his current position to select the most suitable offer.</p>
Related Patterns	<p>PRI_PSA »Privacy Sensitive Architecture«</p> <p>PRI_BPD »Blurred Personal Data«</p>

system; if there is a mismatch, the device may either negotiate acceptable policy adaptations, or it may refuse to participate altogether.

References

The *Cricket* location-support system [23] uses a combination of ultrasonic and RF beacons. The listener uses the time difference between the receipt of the RF signal and the corresponding ultrasonic signal to determine the distance to the beacon sender with high accuracy.

3.3.6 Pattern »Trusted Personal Device« (PRI_TPD)

Without prior establishment of a trust relation between user and Aml environment it is risky to disclose critical data to the system, or to let the system perform sensitive operations on behalf of the user. In order to stay in control, it is advisable for the user not to release personal information to the system unless active system participation in the service provisioning is unavoidable. The PRI_TPD pattern (see p. 31) suggests to confine critical data and computations to a trusted, personal device under exclusive user control.

Synonyms

In a more data-centric context, trusted personal devices are sometimes also called »data vault«; in the context of electronic payment, the term »electronic wallet« has been coined.

Static structure

In a conventional architecture, the trusted device may act simply as a representative for the user, exercising full control of the system on behalf of the user. A more participatory style of interaction would be a trusted device that provides only hints and suggestions to an independent infrastructure. The latter approach is probably better in line with the Aml paradigm, but restricts the user's ability to enforce his privacy policies.

Consequences

- *availability, confidentiality*: A wearable personal device is susceptible to loss, theft, or damage. It is a challenge to combine autonomous operation under normal conditions with data protection in case an adversary gets hold of the device. The more confidential data is maintained on the portable device, the more critical is physical protection of that device.

Trusted Personal Device PRI_TPD A user may be unwilling to entrust his private data to an Aml system with unknown or dubious privacy policies. This pattern suggest to confine critical data and computations to a trusted, personal device under exclusive user control.	
Background	Without prior establishment of a trust relation between user and Aml environment it is risky to disclose critical data to the system, or to let the system perform sensitive operations on behalf of the user. In order to stay in control, it is advisable for the user not to release personal information to the system unless active system participation in the service provisioning is unavoidable.
Problem	How can we avoid to put too much trust in the integrity and discretion of an Aml environment?
Solution	<p>The user should employ a trusted personal device (e.g., a PDA or a mobile phone) to query, store, process, and present sensitive information. The trusted device could store sensitive private data (e.g., medical records), secret keys, and credentials (e.g., digital certificates) on behalf of the user. Encrypting, signing or signature verification could be performed by the device to avoid disclosing cryptographic secrets to the environment. Furthermore, the device may serve as trusted input device for PINs or passwords to protect the user from key grabbing and similar attacks, and critical output should only appear on the user-controlled display. The trusted device may also blur information (cf. »Blurred Personal Data« on page 25) according to predefined privacy rules before revealing it to the environment. Finally, the trusted device can collect evidence for critical system transactions in a secure audit trail that ensures non-repudiation if a dispute about privacy violations should occur.</p> <p>A trusted personal device is also a prerequisite for the implementation of the »Beacon« pattern (cf. p. 29). The user may use the personal device for semiautomatic enforcement of his specified privacy preferences.</p>
Issues	It is essential that we can really trust our personal device. Therefore, it must be protected against subversion (e.g., malware). As the device is likely to hold sensitive information, adequate protection of the data in case of loss or theft is critical. Reconciling the needs of permanent availability (i.e., autonomous processing without explicit user authorization for each new transaction) and protection against unauthorized data access is a difficult design problem.
Examples	<p>A smartcard is often used as a trusted device to apply signatures to transactions on behalf of the user. As the private signature key never leaves the smartcard, an environment can never sign data without smartcard cooperation.</p> <p>A PDA may use a so-called »zero knowledge proof« to prove that the user is in possession of a certain credential without revealing a single bit of the credential itself. This mechanism may be used for unlinkable transactions such as anonymous payment.</p> <p>A PDA may provide private data (e.g., the user's current position) at different levels of accuracy depending on the needs and trustworthiness of the service requesting the data.</p> <p>A PDA may serve as an RFID Guardian [24] that mediates trusted RDIF queries or selectively emulates RFID tags according to a user-defined privacy policy, blocks hostile RFID scans with selective RFID jamming, monitors RFID reader scans in the environment, and detects all hidden tags within radio range to warn the user about goods that may be trackable.</p>
Related Patterns	PRI_PSA »Privacy Sensitive Architecture« PRI_BPD »Blurred Personal Data«

- *availability, performance*: If too much tasks are transferred to the personal device, this might overtax the device's resources, causing slow interaction, service degradation, memory exhaustion, or energy depletion. The stationary environment of the user's mobile device has probably better and more reliable resources.
- *maintainability, integrity*: Mobile personal devices are notoriously hard to maintain, and it is difficult to reliably backup their local data.
- *usability, cost*: If an application requires a personal trusted device, this entails substantial up-front investment for the user with respect to equipment and proper initialization. This prerequisite may deter potential users from participation, or it may even exclude user who cannot afford such an investment.

Variants

Instead of trusting a portable device, the user may trust a remote server to store personal data, to make it available on request, and to enforce pre-define privacy and sharing rules when exchanging personal data with other individuals. An architecture along these lines is the »infospace« model proposed in [10], amounting to network-addressable, secure tuple spaces hosted on trusted servers. In this framework, only a limited amount of data is kept on the personal device. Instead, sensors and mobile devices try to push all relevant data as soon as possible to the personal infospace of the user. From there the information is made available to the system and to other users according to individual security policies. A trusted server has more and better resources to protect the user's privacy; the downside is that this model requires more stable communication links and severely limits the user's autonomy once he becomes disconnected from his infospace.

References

In [22] Pisko et al. give an account of the relevant players, usage scenarios, and the options mobile trusted computing can offer.

3.3.7 Pattern »Physical Privacy Zone« (PRI_PPZ)

People need places where they feel free from being tracked or monitored, and where they can take shelter from unsolicited information overload. The Fair Information Practices expressly mandate information shelter. As a prerequisite, the PRI_PPZ pattern (see p. 33) suggests to assign intuitive privacy assurance attributes to physical locations, allowing people to know what areas can or cannot be considered »privacy safe«.

Physical Privacy Zone PRI_PPZ This pattern assigns intuitive privacy assurance attributes to physical locations, allowing people to know what areas can or cannot be considered »privacy safe«.	
Background	People need places where they feel free from being tracked or monitored, and where they can take shelter from unsolicited information overload.
Problem	How can people know in what areas information is being collected about them, and what privacy policies apply to the collected data?
Solution	Divide the physical Aml environment in distinct privacy zones with intuitive privacy policies that are clearly communicated. Letting users know what information about them will be captured and what information may be brought to their notice is a step towards a socially acceptable Aml environment. Users who enter an insecure privacy zone might consider being more discreet or conservative about their behavior, as in any normal public setting.
Issues	It might be desirable that a privacy zone enforces different policies for different users. Unfortunately, not all sensors can differentiate between legitimate target individuals and »non-combatants« to be exempted from surveillance. For example, audiovisual surveillance monitors all inmates alike. It is almost impossible to communicate intricate privacy policies for a privacy zone for the casual user. Therefore, policies attributed to a physical privacy zone must be either straightforward and intuitive, or lend themselves for automated processing.
Examples	The user should be safe in assuming that the restrooms are excluded from video surveillance. As another example, a user might implicitly understand that information captured by a sensor network in a corporate building will not be disseminated outside the corporate environment. Many hospitals that use locator systems for finding doctors and nurses do not have installed the systems in cafeterias, lounges, or outside the building; this allows personnel not to be found if they are taking a break or doing private activities. The default dissemination policy for data captured inside a user's home should be to keep the information within the home's boundaries. Any data transfer to a third party should be prevented in the absence of explicit user consent.
Related Patterns	PRI_PSA »Privacy Sensitive Architecture« PRI_INV »Invisibility Mode«

Consequences

- *availability*: A physical privacy zone that is completely sealed off the Aml environment will make the user completely »invisible« for the system. Unfortunately, the user is even disconnected from emergency calls and other important notifications that take priority over the users privacy needs.
- *integrity*: Invisible occupants may cause inconsistent global state in an Aml environment, leading to a wrong situation assessment or poor situation awareness.
- *privacy*: It might be desirable that a privacy zone enforces different policies for different users. For example, a household equipped with assisted living facilities will probably capture some behavioral or health status data of the assisted person. However, a visitor of the assisted person should be excluded from such monitoring, that is, the household should be a physical privacy zone from the visitor's perspective. Unfortunately, not all sensors can differentiate between legitimate target individuals and »non-combatants« to be exempted from surveillance. For example, audiovisual surveillance monitors *all* inmates alike.
- *usability*: It is almost impossible to communicate intricate privacy policies for a privacy zone for the casual user. Therefore, policies attributed to a physical privacy zone must be straightforward and intuitive unless they are available in machine-readable form that can be processed by a »Trusted Personal Device« (see p. 31).

Variants

A »Physical Privacy Zone« may constantly advertise its privacy policy (or a link to the policy) so that a user — or rather: the user's »Trusted Personal Device« — can check whether the zone matches her preferences. Ideally, policy information should be provided with some well-known, universally applied mechanism. This would be in line with the Fair Information Practices' principles of *transparency* and *notification*.

3.3.8 Pattern »Invisibility Mode« (PRI_INV)

People do not always want to be found. It can be for many reasons, from being busy and not wanting any distractions, to experiencing strains in relationships, or being victims of stalking attacks. The PRI_INV pattern (see p. 35) suggests means that support users in not being traceable, and not being molested by others.

Invisibility Mode PRI_INV	
This pattern aims at not being traceable, and not being molested by others.	
Background	People do not always want to be found. It can be for many reasons, from being busy and not wanting any distractions, to experiencing strains in relationships, or being victims of stalking attacks.
Problem	Is there a simple way for the user to control whether data about a user is revealed to other users, or whether their requests and data streams can reach her?
Solution	Provide an invisible mode that lets users stop the flow of data to others. In some cases, the invisible mode makes it look like you are not connected or not present, when in reality you actually are. That is, queries and notifications will be forwarded and brought to attention, but receipt is not acknowledged. This pattern may be combined with the »Physical Privacy Zone« pattern (p. 33) to provide an information shelter.
Issues	Some Aml systems are only useful if a certain minimum amount of sensor data is available for a critical mass of users. If too many users decline participation the system may no longer be able to provide meaningful service. To motivate user cooperation, the system must provide clear benefits for its participants.
Examples	The »invisible mode« in many instant messengers make it look like the person is not logged in; however, users can still see others and receive messages. An answering machine enables a user to listen to incoming telephone calls without revealing her presence. If she likes, she can selectively pick up the phone to accept a call by switching from recording mode to interactive mode. Depending on the key by which it is stimulated, an RDIF token may provide different identities, or it may refuse to reply to the stimulus altogether.
Related Patterns	PRI_PSA »Privacy Sensitive Architecture« PRI_PPZ »Physical Privacy Zone«

Synonyms

This pattern is also known as »Invisible Mode« or »Anonymous Mode«.

Consequences

- *privacy*: An »Invisibility Mode« provides conditional, selective privacy to the user's discretion. If the design aims at unconditional »invisibility«, then the »Physical Privacy Zone« pattern is preferable.
- *usability*: This pattern is likely to require a significant amount of user participation because privacy preferences must be specified. It is quite difficult to define personal privacy needs unambiguously and correctly.
- *accountability, usability*: In some situations it might be embarrassing for the user if other users can find out that he is running »Invisibility Mode« (e.g., it may be considered impolite to ignore a connection request). The designer must consider this possibility; in critical situations, »Invisibility Mode« either should be indiscernible for other users, or it should be indistinguishable from simply being disconnected.
- *accountability, usability*: Who is to blame for an ambiguous or inconsistent privacy policy that either causes privacy breaches, or seals off the user from important messages — the user who provided the flawed policy, or the system accepting it without complaint?

Variants

A simplistic solution is to use the »Trusted Personal Device« pattern and to provide privacy by either shielding the physical zone from RF signals (alternatively, RF communication may be jammed to prevent successful data exchange) to establish a »Physical Privacy Zone«, or by asking the user to switch-off the personal device if he requires privacy. The disadvantage of such crude approaches is that the user is fully disconnected, that is, he is unable to selectively receive (or respond to) any incoming requests. To stay in control, the system should continue to provide the user with incoming information — filtered according to some privacy policy — but it should provide (logical) »invisibility« to the user's sole discretion.

3.3.9 Pattern »Privacy Feedback« (PRI_PFB)

The user is not always aware of the privacy policy of his current environment. Even if he is, privacy feedback is required to verify that the system complies to its

Privacy Feedback PRI_PFB	
This pattern provides information to the user about privacy-relevant events and conditions.	
Background	The user is not always aware of the privacy policy of his current environment. Even if he is, privacy feedback is required to verify that the system complies to its policies, to hold the system responsible for any policy violations, and to provide the user with an opportunity to opt-out whenever he feels a disproportion between the potential gain in quality of service and the loss in privacy this would entail.
Problem	One of the goals of ubiquitous computing is to make computers invisible and unremarkable, making them disappear into the background of everyday life. However, if a system is invisible, how can users understand what personal information is being collected and where that information flows?
Solution	The system should always display a response to user actions that lead to major state changes, or that have irreversible consequences. There are three opportunities for the system to provide privacy notifications: <ul style="list-style-type: none"> • <i>Before</i> sensitive data is accessed: The system can inform the user that he is about to enter a physical privacy zone that will capture specific personal information — maybe without further notice. (Entering the zone despite of this notification may be interpreted as implicit consent.) • <i>During</i> sensitive data access: A short warning may remind the user that he is under surveillance; the user may adapt to the situation by behaving more discreetly. A notification during access may be combined with an opt-out opportunity for the user. • <i>After</i> sensitive data access: Sensitive access may be logged so that although access cannot be prevented after the fact, the user can still hold the system (or other users of the system) liable for any unauthorized privacy intrusion.
Issues	A key design decision here is whether the user is simply notified or has a choice over whether information is disclosed or not. There are plausible cases for each. There is a trade-off concerning the level of trust the user wants to put into the system and its other users on the one hand, and the level of user participation the user is willing to accept on each critical transaction. Excessive privacy feedback can turn into a nuisance that provokes bypass and disregard. The system should rather promote a <i>possibility</i> than an <i>obligation</i> for privacy feedback.
Examples	A trusted personal device (cf. to page 31) could serve as a »privacy mirror« [18, 5], reflecting to the user what types of information the environment is monitoring (e.g., audio, video, movement, identity), how it is monitoring the user (e.g., tracking his RFID token), and what the system currently knows about him. For unobtrusive feedback, the personal device may simply depict this information with a set of intuitively understood icons. In an emergency situation, a paramedic may not be able to wait for user consent to access the patient's critical medical records. However, such unauthorized access can be recorded in an audit trail so that the patient has an opportunity to challenge the need of this privacy intrusion after it has occurred, and to hold the caregiver liable if access was not really justified. AT&T Wireless »Find Friends« service notifies your friend if you ask for his or her location; the friend can accept the request or deny it and remain »invisible« (see »Invisibility Mode« pattern on p. 35).
Related Patterns	PRI_PSA »Privacy Sensitive Architecture«

policies, to hold the system responsible for any policy violations, and to provide the user with an opportunity to opt-out whenever he feels a disproportion between the potential gain in quality of service and the loss in privacy this would entail. The PRI_PFB pattern (see p. 37) suggests different means to provide sufficient information to the user about privacy-relevant events and conditions.

Consequences

- *accountability, availability, integrity*: If the system provides privacy feedback, the feedback must be reliable, accurate, and up-to-date. It can be quite challenging and costly to provide adequate feedback according to the user's expectations under all circumstances.
- *usability*: The designer must carefully consider the trade-offs between privacy awareness and cognitive overload. Privacy implications must be displayed in simple terms, and not too frequently. Furthermore, feedback must be adequately adjusted to the situation (e.g., it must be reduced to a minimum in potentially hazardous situations to avoid user distraction).
- *performance*: Feedback may slow down system/user interaction. The performance impact is even stronger if feedback is combined with choice.

References

In [18] Mynatt and Nguyen argue that ubiquitous computing should be made visible for the user (in contrast to the prevalent ideal of invisibility) and they propose so-called »privacy mirrors« for simple and effective privacy feedback.

3.3.10 Pattern »Limited Data Retention« (PRI_LDR)

Private data must be protected against unauthorized disclosure, and it must be kept accurate, complete, relevant, and valid. The fewer data is kept, the easier it is to satisfy these requirements.

Furthermore, the Fair Information Practices postulate data parsimony and legitimation for volume as well as retention period of personal data that has been collected. The PRI_LDR pattern (see p. 39) suggests some strategies for limiting the retention period of personal data.

Consequences

- *accountability, availability*: There may be legal obligations for long-term data retention (e.g., in HIPAA- or FDA-regulated environments).

Limited Data Retention PRI_LDR	
Only a minimal amount of private data should be kept in order to alleviate the burden of complying with the Fair Information Practices.	
Background	Private data must be protected against unauthorized disclosure, and it must be kept accurate, complete, relevant, and valid. The fewer data is kept, the easier it is to satisfy these requirements.
Problem	How can we minimize the risks of unauthorized disclosure of personal data?
Solution	One approach for collecting sensitive data is to keep that data only for a limited period of time. Expiry of personal data should be the default option. Typical strategies are <ul style="list-style-type: none"> • to remove all data older than a predefined retention period, or • to use a cyclic buffer, storing only the n most recent updates; for $n = 1$ this amounts to storing only the last known entry.
Issues	There may be legal obligations for long-term data retention (e.g., in HIPAA- or FDA-regulated environments). A limited retention period constrains computer forensics if the system suffers an attack (e.g., a computer virus, or unauthorized access).
Examples	A system could keep track of a person's location only for the past 8 hours. This is similar to how mobile phones only keep a limited number of entries in the call history.
Related Patterns	

- *accountability*: A limited retention period constrains computer forensics if the system suffers an attack (e.g., a computer virus, or unauthorized access)
- *usability*: Reducing the amount of available data may degrade service quality by affecting the utility of the remaining data.
- *performance, cost*: A shorter retention period avoids information clutter and should be beneficial for performance and cost of data retention. However, to draw actual profit, overdue data must be identified and deleted automatically, without costly manual intervention.

4 Establishing Trust

A system is only secure if the user *can* trust it. However, that does not imply that the user actually *will* trust a secure system. Therefore, it is not sufficient to provide a secure design, but it is equally important to convince the users that they can put trust in the availability, integrity, privacy, and — ultimately — loyalty of the system: If the system is not trustworthy, it is not usable! This is particularly true for Aml systems that are covert by design, and thus even more susceptible of misconduct.

4.1 Factors contributing to trust

A number of factors affect the user's willingness to trust a system. There are factors that can be controlled — or at least positively influenced — with pure technology. Others originate from the user's fundamental mindset and long-term convictions; these are particularly hard to change.

Individual disposition to trust

People have a general ability to trust that forms a kind of baseline attitude when they approach any trust situation, and some people have a higher baseline level of trust than others [2]. This basic trust is a relatively stable personality characteristic, and is one of the factors that contribute when making decisions about trust. Experiments have shown that high and low trusters had markedly different opinions and behaviors.

Individual risk perception

Similar to basic trust, users may have a basic or baseline level of perceived risk. This is probably best described as a bias to perceive situations as being risky or risk free. The more risky users perceive a system or a particular system interaction, the less they are inclined to trust the system.

Actual risk

The more personal values (e.g., private information, money, irreversible decisions) are at stake, the higher is the perceived risk — provided, of course, the

user is able to recognize the true consequences of his system interaction. The system should split operation into smaller transactions with manageable risks.

Uncertainty

Users prefer predictable systems. The more they are surprised by the system's appearance or behavior, the less they trust the system.

Choice

Users do not want to be blackmailed by the system. A system is not perceived as particularly trustworthy if it forces the user into a certain interaction pattern without providing alternative options, or if users that refuse to accept the system's single offer for interaction are unduly discriminated. For example, an online shop that offers only a single payment method (if anything, some obscure homemade method requiring detailed user registration) is more suspicious than a competitor that accepts credit cards, cash checks, or bank transfer alike.

Device autonomy

Users fear paternalism. They mistrust system agents that act too autonomously, and they prefer to participate in — or even to control — system behavior. This general disposition, however, runs counter to smart and unintrusive user support.

Feedback

Users want to know what is going on. They expect the system to clearly communicate relevant state changes and activities that might affect them.

Experience

Trust grows with increasing positive experience. Users tend to probe the system, to put it to the test in minor cases before they gradually gain trust. It takes time to appreciate the trustworthiness of a system. However, trust may be transferred to speed up trust establishment: user Alice might simply decide to follow the recommendation of the more experienced user Bob, provided that Alice trusts Bob.



From the designer's perspective, we can try to provide technical support to minimize the actual risks, to keep the user informed, to offer different options for system interaction, to keep the user involved, and to enable persistent positive trust experiences. The only way to affect the user's general trust disposition is to persistently meet the user's trust expectations at the technical and organizational level.

4.2 Design patterns that promote trust

Below we discuss design attributes promote trust in the functionality and security of the system.

4.2.1 Pattern »Trust Enhancing Architecture« (TR_TEA)

If a system is expected to handle personal data, establishing trust is a prerequisite for usability. Trust establishment is particularly important for Aml systems, which tend to collect a significant amount of personal data while aiming at autonomous, smart processing with minimal obtrusiveness and user involvement. The TR_TEA pattern (see p. 44) provides guidance on technical solutions that contribute to trust.

Consequences

- *accountability*: Providing comprehensive feedback and offering choice puts the user back into control, relieving the system of full accountability.
- *availability, confidentiality, integrity*: Striving for these classic security virtues goes hand in hand with promoting trust.
- *usability*: There is a conflict between steady user involvement in trust-related decisions on the one hand, and unintrusiveness of service on the other. There is also a conflict between the system's quest for intelligent user support and the user's fear of paternalism.
- *performance, costs*: Enabling user participation and user choice will slow down service provisioning, and it may lead to a more expensive system design.

References

Chapter 9 of PISA's Privacy Handbook [2] discusses factors that contribute to trust; it contains references to further reading.

Trust Enhancing Architecture TR_TEA This patterns suggests general design features that encourage users to trust the system.	
Background	If a system is expected to handle personal data, establishing trust is a prerequisite for usability. Trust establishment is particularly important for Aml systems, which tend to collect a significant amount of personal data while aiming at autonomous, smart processing with minimal obtrusiveness and user involvement.
Problem	How can the system promote trust establishment at a technological level?
Solution	Transparency is the system's most fundamental contribution that enables trust relations. Transparency has two main aspects: 1. Make the system comprehensible to avoid uncertainty 2. Make the system predictable to avoid surprise and to nurture confidence. Design can contribute to transparency in many ways: <ul style="list-style-type: none"> • Provide a simple mental model for the way the system works. users will develop mental models and assumptions about the system even when no information is provided, and these models may be wrong. To prevent this, developers should explicitly guide the model development by showing the operation of the system. • Use proven patterns of interaction, that is, follow established manual procedures and existing conventional solutions from related domains to foster understanding by analogy. • Strive for predictable performance with respect to consistency of service provision and user feedback, consistent look and feel of interfaces, and even system response times. • Provide comprehensive information about system status and state of user-related processing, as well as about options for system interaction (or for opt-out, respectively). • Foster perceived stability, that is, avoid unexpected configuration changes or down-times. • Reduce the user's risks by following the principles of data parsimony, limited data retention period, encryption of sensitive data, and other privacy enhancing measures.
Issues	Unfortunately, many aspects of the TR_TEA pattern run counter to the Aml ideal of invisible, smart computing without any explicit user interfaces.
Examples	Many Web applications use so-called cookies to store persistent information on the client's browser such as session state, user credentials, or usage patterns. Unfortunately, cookie processing is often not visible to the user, and therefore users tend to mistrust cookies. To improve transparency, many browsers provide functionality to list what cookies are stored on a system, and an ability to view the information stored within them. An optional pop-up box can notify the user about servers wishing to store a cookie on the local browser. Many services further promote trust by providing reasonable service even if the user refuses to accept cookies — maybe with slightly degraded quality, user-friendliness, or usability.
Related Patterns	PRI_PSA »Privacy Sensitive Architecture« PRI_PET »Privacy Enhancing Technology«

5 Summary and Outlook

The security challenges that are most characteristic of Aml systems revolve around privacy and trust. Consequently, most patterns compiled in this report address privacy or — more abstractly — confidentiality issues. That does not mean that the other classic virtues of security (integrity, availability, accountability) are no longer relevant in Aml environments: it is just that the corresponding design solutions — if any — tend to boil down to the standard patterns known from conventional IT [27].⁴

Concerning privacy and trust, we pointed out the inherent conflict between unintrusive, proactive, and »intelligent« service provision on the one hand, and comprehensive user control on the other. Although we expect further progress in finding technical means to reconcile these conflicting goals, we are rather sceptical whether the customary concepts of privacy are applicable to Aml systems without modification. Many researchers share our scepticism. In allusion to Samuel Warren and Louis Brandeis, who wrote the influential paper *The Right to Privacy*, motivated largely by the advent of modern photography and the printing press, Langheinrich wrote in [17]:

As important as it is to take existing laws and codes of practices into account, which can and must serve as important guidelines for creating privacy-respecting infrastructures it is equally important to remember that laws can only work together with the social and technological reality, not against them. If certain legal requirements are simply not enforceable, technological or procedural solutions need to be found, or the law changed.

Maybe it is indeed time that we face the new technological realities and accept the fact that personal data collection will continue to advance and erode privacy as we know today. But new paradigms will take place of old and unrealistic assumptions, and new forms of human interactions will evolve in society, just as we have learned to live with the specters (i.e., modern photography) that haunted Warren and Brandeis more than 100 years ago.

Time will tell to what extent users are willing to scale down their security and privacy expectations to fully exploit the potential of Aml technology, while still standing up for their fundamental right to privacy.

⁴ Note, though, that many conventional security design patterns do not fit the design constraints of typical Aml systems. Therefore, we still lack convincing solutions for many security issues raised by the Aml paradigm.

Summary and Outlook

References

- [1] Bob Blakley, Craig Heath et. al: Security Design Patterns. Technical Guide G031, The Open Group, April 2004
ISBN 1-931624-27-5
<http://www.opengroup.org/publications/catalog/g031.htm>
- [2] G.W. van Blarkom, J.J. Borking, and J.G.E. Olk (eds.): Handbook of Privacy and Privacy-Enhancing Technologies — The case of Intelligent Software Agents. PISA Consortium, College Bescherming Persoonsgegevens, The Hague, 2003
ISBN 90 74087 33 7
<http://www.andrewpatrick.ca/pisa/handbook/handbook.html>
- [3] A. Bullock and S. Benford: An access control framework for multi-user collaborative environments. Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work (GROUP'99, Phoenix, Arizona), pp. 140–149, November 1999
ISBN1-58113-065-1
http://portal.acm.org/ft_gateway.cfm?id=320313&type=pdf&coll=GUIDE&dl=GUIDE&CFID=12602&CFTOKEN=67947330
- [4] J. Camenisch and E. Van Herreweghen: Design and Implementation of the Idemix Anonymous Credential System. Proc. 9th ACM Conference on Computer and Communications Security (Washington DC, USA), pp. 21–30, November 2002
<http://www.zurich.ibm.com/security/idemix/>
- [5] E. Chung et al.: Development and Evaluation of Emerging Design Patterns for Ubiquitous Computing. Proceedings of Designing Interactive Systems DIS2004 (Cambridge, Massachusetts), pp. 233–242, August 2004.
<http://guir.berkeley.edu/projects/patterns/>
- [6] EC: Directive 95/46/EC on the protection of individuals with regard to the processing of personal data and on the free movement of such data. European Parliament and Council, October 1995
http://ec.europa.eu/justice_home/fsj/privacy/law/index_en.htm
- [7] EC: Directive 2002/58/EC concerning the processing of personal data and the protection of privacy in the electronic communications sector (Directive on privacy and electronic communications). European Parliament and Council, July 2002
http://ec.europa.eu/justice_home/fsj/privacy/law/index_en.htm

- [8] E. Gamma, R. Helm, R. Johnson, and J. Vlissides: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995
- [9] HHS: HIPAA Administrative Simplification Regulation Text, 45 CFR Parts 160, 162, and 164. Unofficial version, as amended through February 16, 2006, U.S. Department of Health and Human Services, February 2006
<http://www.hhs.gov/ocr/AdminSimpRegText.pdf>
- [10] J.I Hong and J.A. Landay: An Architecture for Privacy-Sensitive Ubiquitous Computing. Proc. 2nd International Conference on Mobile Systems, Applications, and Services (MobiSys 2004), Boston, Massachusetts, Boston, Massachusetts, pp. 177–189, June 2004
http://portal.acm.org/citation.cfm?id=990087&dl=GUIDE&coll=GUIDE&CFID=432311&CF_TOKEN=62131560
- [11] D.M. Kienzle, M.C. Elder, D.S. Tyree, and J. Edwards-Hewitt: Security Patterns Template and Tutorial. February 2002
http://www.modsecurity.org/archive/securitypatterns/dmdj_template_and_tutorial.pdf
- [12] D.M. Kienzle, M.C. Elder, D. Tyree, and J. Edwards-Hewitt: Security Patterns Repository Version 1.0. 2002
http://www.modsecurity.org/archive/securitypatterns/dmdj_repository.pdf#search=%22%22Security%20Patterns%20Repository%22%20Kienzle%22
- [13] S. Konrad, B.H.C. Cheng, L.A. Campbell, and R. Wassermann: Using Security Patterns to Analyze Security Requirements. Second International Workshop on Requirements Engineering for High Assurance Systems (RHAS03), Monterey Bay, California, September 2003
<http://ftp.cse.msu.edu/~konradsa/Publications/rhas03.pdf>
- [14] S. Lahlou and F. Jegou: European Disappearing Computer Privacy Design Guidelines, Version 1.1. European Community IST/Disappearing Computer Initiative, Ambient Agoras programme (IST-2000-25134), October 2004
<http://www.rufae.net/privacy.html>
- [15] S. Lahlou, M. Langheinrich, and C. Röcker: Privacy and Trust Issues with Invisible Computers. Communications of the ACM, Vol. 48, No. 3, pp. 59–60, March 2005
<http://portal.acm.org/citation.cfm?id=1047705>
- [16] M. Langheinrich: Personal Privacy in Ubiquitous Computing - Tools and System Support. PhD thesis No. 16100, ETH Zurich, Zurich, Switzerland, May 2005
<http://www.vs.inf.ethz.ch/publ/papers/langheinrich-phd-2005.pdf>
<http://www.vs.inf.ethz.ch/publ/papers/langheinrich-phd-2005.talk.pdf>
- [17] M. Langheinrich: Privacy by Design — Principles of Privacy-Aware Ubiquitous Systems. Proc. Third International Conference on Ubiquitous Computing (UbiComp 2001, Atlanta, USA), LNCS No. 2201, Springer-Verlag, pp. 273–291, 2001
<http://www.vs.inf.ethz.ch/publ/papers/privacy-principles.pdf>

- [18] E.D. Mynatt and D. Nguyen: Making Ubiquitous Computing Visible. Position paper, Workshop on Building the Ubiquitous Computing User Experience, ACM Conference on Human Factors in Computing Systems (CHI), Seattle, Washington, April 2001.
<http://www.erstwhile.org/writings/chi2001Position.pdf>
- [19] Thiên-Lộc Nguyễn: National Identification Systems. Master Thesis, Massachusetts Institute of Technology, June 2003
<http://theory.lcs.mit.edu/~cis/theses/nguyentl-masters.pdf>
- [20] The Open Group Security Forum.
<http://www.opengroup.org/security/gsp.htm>
- [21] OECD: OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data. Organisation for Economic Co-operation and Development, 2001
<http://www1.oecd.org/publications/e-book/9302011E.PDF>
- [22] E. Pisko, K. Rannenber, and H. Roßnagel: Trusted Computing in Mobile Platforms. *Datenschutz und Datensicherheit* 29 (2005) 9, pp. 526–530, September 2005
<http://www.wiiv.de/publikationen/TrustedComputinginMobilePlatfo1479.pdf>
- [23] N.B. Priyantha, A. Chakraborty, and H. Balakrishnan: The Cricket Location-Support System. Proc. 6th ACM International Conference on Mobile Computing and Networking (ACM MOBICOM), Boston, MA, August 2000
<http://nms.lcs.mit.edu/papers/cricket.html>
- [24] M.R. Rieback, B. Crispo, and A.S. Tanenbaum: RFID Guardian: A Battery-Powered Mobile Device for RFID Privacy Management. Proc. 10th Australasian Conference (ACISP 2005), Brisbane, Australia, July 2005, Lecture Notes in Computer Science 3574, pp. 184–194, Springer-Verlag 2005
ISBN 978-3-540-26547-4
<http://www.rfidguardian.org/papers/acisp.05.pdf>
- [25] T.S. Saponas, M.K. Prabaker, G.D. Abowd, and J.A. Landay: The Impact of Pre-Patterns on the Design of Digital Home Applications. Proceedings of the 6th ACM Conference on Designing Interactive Systems (DIS), University Park, Pennsylvania, pp. 189–198, June 2006
ISBN: 1-59593-367-0
<http://portal.acm.org/citation.cfm?doid=1142405.1142436>
<http://dub.washington.edu/projects/dpdh/>
- [26] SecurityPatterns.org
<http://www.securitypatterns.org>
- [27] M. Schuhmacher et al.: Security Patterns — Integrating Security and Systems Engineering. John Wiley & Sons, 2006
ISBN 0-470-85884-2

- [28] R. Schwarz and K. Simon: A Security-enabled BelAml Architecture — Security Threats and Proposed Safeguards. Report No. 92.06/E, Fraunhofer IESE, Kaiserslautern, August 2006
- [29] R. Schwarz: Systematic Elicitation of Security Requirements in the Context of Aml Systems. IESE Report No. 98.05/E (BelAml Report No. 3.05/E), Fraunhofer IESE, Kaiserslautern, November 2005
http://bib.iese.fhg.de/reports/public/2005/iese-098_05.pdf
- [30] F. Stajano and R. Anderson: The Resurrecting Duckling: Security Issues for Ubiquitous Computing. Proceedings 7th International Workshop on Security Protocols, Lecture Notes In Computer Science 1796, pp. 172–182, Springer-Verlag 2000
ISBN 3-540-67381-4
<http://www.cl.cam.ac.uk/~rja14/duckling.pdf>
- [31] F. Stajano: The Resurrecting Duckling — what next? Proceedings 8th International Workshop on Security Protocols, Lecture Notes In Computer Science 2133, pp. 204–211, Springer-Verlag 2001
ISBN 3-540-42566-7
<http://www.cl.cam.ac.uk/~fms27/papers/duckling-what-next.pdf>
- [32] F. Stajano: Security for Ubiquitous Computing. John Wiley and Sons 2002
ISBN 0-470-84493-0
<http://www.cl.cam.ac.uk/~fms27/secubicomp/>
- [33] C. Steel, R. Nagappan, and R. Lai: Core Security Patterns. Prentice Hall PTR 2005
ISBN 0-13-146307-1
<http://coresecuritypatterns.com>
- [34] SPC: Break-Glass — An Approach to Granting Emergency Access to Healthcare Systems. Whitepaper, Joint NEMA/COCIR/JIRA Security and Privacy Committee (SPC), December 2004
<http://www.nema.org/prod/med/security/>
- [35] W. Tolone, G.-J. Ahn, and T. Pai: Access Control in Collaborative Systems. ACM Computing Surveys, Vol. 37, No. 1, pp. 29 – 41, March 2005
<http://portal.acm.org/citation.cfm?id=1057977.1057979>

List of Abbreviations

AAA	Authentication, Authorization, and Access Control
Aml	Ambient Intelligence
COCIR	European Coordination Committee of the Radiological and Electro-medical Industry
DoS	Denial of Service
EC	European Community
GPS	Global Positioning System
GUIR	Group for User Interface Research (University of California at Berkeley)
HHS	U.S. Department of Health and Human Services
HIPPA	Health Insurance Portability and Accountability Act (1996)
IT	Information Technology
JIRA	Japan Industries Association of Radiological Systems
NEMA	National Electrical Manufacturers Association (USA)
OECD	Organisation for Economic Co-operation and Development
PawS	Privacy-aware System
PDA	Personal Digital Assistant
PET	Privacy Enhancing Technology
PIN	Personal Identification Number
PSTN	Public Switched Telephone Network
RF	Radio Frequency
SPC	Joint NEMA/COCIR/JIRA Security and Privacy Committee

Document Information

Title: Security Design Patterns for Ambient Systems
IESE Report: 115.06/E
Date: October 2006
Version: 1.0
Status: final
Distribution: public

Copyright 2006 Fraunhofer IESE and TU Kaiserslautern.
All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means including, without limitation, photocopying, recording, or otherwise, without the prior written permission of the publisher. Written permission is not needed if this publication is distributed for non-commercial purposes.

