Towards a History–Based Authentication and Intrusion Detection Mechanism

José Carlos Ciria , Eladio Domínguez, Inés Escario, Ángel Francés, María Jesús Lapeña, María A. Zapata[a]

[a] *Computer Science and Engineering Systems Department, Zaragoza University, Spain*

Abstract

We propose a history-based mechanism designed to strengthen the authentication and intrusion detection processes. It can be applied to entities endowed with communication, memory and processing capabilities that enable them to record, recognise and report on their history. The mechanism is supported by a pattern that provides the notions of *history* and its atomic component, *occurrence*. Our proposal is test-bedded in a security and safety system, framed within the context of smart environments, designed for the protection of personnel and facilities and responsible for its self-protection and self-monitoring.

Keywords:
authentication, intrusion detection, security pattern

1. Introduction

The consolidation of increasingly complex communication infrastructures and the development of artificial entities enhanced with ever-growing capabilities (such as sensing, communication, memory and processing) enable environments of increasing sophistication. The most global of such environments is the Internet of Things, IoT, where electronic devices are embedded into everyday physical objects, so that real and virtual worlds seamlessly integrate within the resulting cyberphysical infrastructure (Miorandi et al., 2012). Among the burgeoning fauna populating these environments, one can find entities associated with humans to enhance their capabilities (e.g., smart-cards), smart-objects (Atzori et al., 2010), robots, bots and avatars (Gavrilova and Yampolskiy, 2010), cognitive robots and cognitive computers (Wang, 2010) or spimes (Sterling, 2005).

As a result, it is natural to envision complex systems that can provide critical services (such as security or logistics management) with reduced or no human intervention. The management of such systems poses challenges, not the least of which is to assure the identity of the entities supplying crucial services or accessing sensitive information and resources.

In this paper, we focus on the issues of entity authentication and intrusion detection. Authentication, defined as proving that an entity (human or not) is what he/it claims to be, relies on three types of authentication factors: something the claimant knows (e.g., a PIN, a password), something the claimant has (e.g., an ID-badge, a cryptographic key, a smart-card) or something the claimant is (biometric-based authentication, based on an individual's physical and/or behavioural traits or, in the case of non-human entities, secret keys assigned to hardware components) (Burr et al., 2011). None of these factors is free from concerns, some of which are summarised below:

- What the claimant knows may be guessed by an attacker or discovered by eavesdropping.
- What the claimant has may be lost, damaged, stolen or cloned. To have deeper insight into the weaknesses of particular solutions, such as smart-cards, we refer the reader to (Madhusudhan and Mittal, 2012).
- Biometric-like features can be defined in non-human entities. These features include PUFs (Physical Unclonable Functions) (Suh and Devadas, 2007), radiofrequency certificates of authenticity (RF-COAs) and radiofrequency distinct native attributes (RF-DNA) (Cobb et al., 2012), that depend on complex physical characteristics of integrated circuits and differ from chip to chip. Though promising, these solutions are not yet mature or widely applicable.

To summarise, there is no single authentication factor providing a universal solution to the authentication problem. It thus remains an open question.

In this paper, we propose a history-based scheme for authentication and intrusion detection. Our proposal generalises the first factor (what the entity knows) by increasing the complexity of the information an entity must prove to possess to be authenticated. We aim to exploit the enhanced capabilities of artificial entities to simulate the natural intelligence capability of humans. Instead of relying on the knowledge of just a password or a PIN, our proposal

mimics a natural way of verifying one person's identity: interrogating him on his past, exploiting the available knowledge about his history.

Our proposal incorporates features such as challenge-response mechanisms (where an entity must successfully respond to a series of challenges) and identity verification through multiple proofs of identity (Adi, 2010; Paci et al., 2009; Bhargav-Spantzel et al., 2010). We consider such features in the framework of the history of the entity to be authenticated. Far from being a more restrictive option, ours can be seen as a generalisation of those solutions, as an entity's history encompasses everything that happens to it: learning the response to a challenge or acquiring the identity attributes belong to an entity's history.

An advantage of our proposal is its dynamism and flexibility, because the number and nature of questions posed to the claimant can be adapted to the criticality of the context. In addition, it allows collaboration, as multiple sources can be consulted to gather a thorough knowledge of the history of a claimant; such a feature is highly relevant in decentralised environments such as the IoT (Miorandi et al., 2012). Moreover, it enables tampering and intrusion detection, because the inability of an entity to successfully account for its own history may be an indication of integrity loss or impersonation. Additionally, it is fully applicable to artificial entities, provided they have enough storage and computational capabilities.

These advantages may render our proposal useful to reinforce authentication processes, either by itself or by complementing other approaches. As a result, it helps to prevent unauthorised disclosure of data and malicious information modification or destruction and to assure the identity of the entity that has performed an action. Thus, the confidentiality, integrity, authenticity and non-repudiation are strengthened (ISO, 2012), (Ross et al., 2008).

The paper is structured as follows: Section 2 explains the intuitive basis of our approach in depth. In Section 3, we put our proposal in context, comparing it with other approaches. Section 4 describes the security pattern that supports our proposal. In Section 5, we describe a security and safety system, designed based on our pattern and developed within the framework of the project THOFU (Consortium, 2010). The last section of the paper presents the conclusions and future directions for work.

2. Our proposed approach

Our proposal is inspired by the natural way in which people authenticate entities: relying upon the available knowledge about the entity that must be authenticated. Let us analyse some examples extracted from real life.

Our first example illustrates an intuitive way of authenticating someone when meeting them for the first time or after a period without contact: asking him/her questions that only the correct person would naturally know the answers to. A celebrated case is that of Martin Guerre, absent from his community from 1548 to 1556, whose identity was disputed by two different persons (Davis, 1983).

Our second example is Himalayan mountaineers who need to prove they have reached a summit. To do that, they subject themselves to an intensive scrutiny by Elizabeth Hawley (Jolly, 2010), who exhaustively interrogates them on the circumstances of their ascent; the right answers to her questions provide a widely recognised (though unofficial) certification of the authenticity of their claim.

In both cases, a system is required to decide on the authenticity of an assertion made by a person. To do that, the system searches its memory of all of the facts regarding that person that may help to confirm the assertion. We remark that the system's memory is collective (a community's memory, information gathered from several sources such as weather reports, or reports coming from other expeditions).

Our proposal consists of providing the verifier with a mechanism for remembering, allowing it to exploit its knowledge about the entity's history to authenticate it. The verifier would be able to increase its knowledge by aggregating information regarding the entity provided by other relied-upon entities. The structure of the inquiry could range from simple question-answer interchange to an exhaustive, extraordinarily complex interrogation (if the entity requires access to a critical resource).

Our mechanism can be applied for different purposes, such as those sketched below.

Demanding an entity to account for its history reinforces the first authentication factor (what an entity knows). Either by itself, or as a complement of other factors, our approach would help in strengthening defence against impersonation.

On the other hand, our mechanism would help in avoiding identification errors because it enables a verifier to check the current appearance, behaviour

or state of an entity (e.g., a patient equipped with a smart bracelet that stores his medical history and registers his vital signs) against those previously recorded by the system.

Moreover, we can envision a surveillance system able to 'perceive' anomalous circumstances, i.e., circumstances departing from the ones kept in its memory (e.g., a too-high temperature or movement detected in a room at night, when it is usually deserted, which may be indications of fire or a robbery).

Intrusion detection systems are usually considered detective safeguards raising a second line of defence once an intruder has circumvented preventive safeguards such as authentication (Biermann et al., 2001; Ye et al., 2002). Our proposed mechanism may provide support to an intrusion detection system, as it allows matching the current behaviour of a user with the one recorded by the system.

3. Related work

Considering its context of application, our proposal is related to research in environments enabled by new technologies, where concerns arise over security (Caire and van der Torre, 2010), (Roman et al., 2011), (Ning and Liu, 2012). Unlike Wireless Sensor Networks, whose nodes are limited in power capacity, computational capabilities and memory (Akyildiz et al., 2002) we focus on systems of entities with enhanced communication, memory and processing capabilities. A context closer to ours would be a system of robots endowed with complex capabilities that face issues such as intrusion detection and the detection of misbehaviours, due to either fault or malice (Bicchi et al., 2010), the detection, identification and determination of behaviours (Ruiz-Del-Solar et al., 2010) or authentication(Gavrilova and Yampolskiy, 2010), (Wang, 2010). Another related context where our proposal could be applied is considered in (González Alonso et al., 2012), where an architecture is proposed to assure the interoperability of service robots and a digital home.

However, the previously cited works propose no specific solutions to deeply examine the authentication problem and consider how to benefit from the enhanced capabilities of the system. While (Gavrilova and Yampolskiy, 2010), (Bicchi et al., 2010) and (Ruiz-Del-Solar et al., 2010) just mention the issue, the authentication mechanisms described in (Wang, 2010) and (González Alonso et al., 2012) are very simple: (Wang, 2010) just verifies a

match between a password communicated by a robot and the one stored in a database. On the other hand, the security and privacy subprotocol presented in (González Alonso et al., 2012) reduces authentication to the verification of an encrypted identifier.

In the context of intrusion detection, for (Bicchi et al., 2010) an intruder robot is a misbehaving agent whose behaviour does not adhere to a set of general rules common to all entities. Our proposal allows an individual monitoring of an entity: the current behaviour of an entity can be benchmarked against its individual historical behaviour.

Our proposal can also be considered in comparison with other specific authentication mechanisms found in the literature. While such proposals have been raised in contexts different than ours, an analysis of their features, challenges and advantages, as well as the requirements imposed on them, provides a basis for validating our proposal.

In particular, our proposal incorporates features from other methods such as challenge-response mechanisms (where an entity must successfully respond to a series of challenges) and identity verification through multiple identity proofs (Adi, 2010), (Paci et al., 2009), (Bhargav-Spantzel et al., 2010). We consider such features in the framework of the history of the entity to be authenticated, as explained below.

(Adi, 2010) proposes a challenge-response mechanism to authenticate an entity's identity: an entity E generates challenge-response pairs randomly and communicates them to a trusted authority. Later, the authority can use these pairs to verify the identity of an entity claiming to be E. Our scheme generalises such mechanisms: questions can be meaningful and refer to occurrences that happened to the entity; instead of relying on information previously provided by the entity, the entity responsible for performing the authentication can use information from its own experience or even gather information from different sources.

In multiproof identity verification schemes, authentication relies on verifying several identity attributes, possibly certified by different identity providers (Paci et al., 2009). A user must prove he possesses one or several identity attributes before he is allowed to access a service. Moreover, the service provider may dynamically choose the required set of identity attributes (Bhargav-Spantzel et al., 2010). Similar to such approaches, ours requires en entity to prove its knowledge based on multiple features to be authenticated; unlike such approaches, in our scheme, virtually any combination of occurrences belonging to an entity's history can be used to authenticate it, and no iden-

tity providers are explicitly required (though they are not excluded). In particular, as issuing and receiving a certificate can be modelled as occurrences happening to an entity and thus part of its history, certified identity attributes and eventual communication with identity providers can be naturally integrated into our proposal.

4. A security pattern for authentication and intrusion detection

In this section, we present a security pattern, named the *history-based authentication pattern*, that provides a structured description of our history-based proposal. Like other authentication patterns (Lee Brown et al., 1999), it aims to reinforce the system security by preventing non-authorised entities from entering the system or accessing its services and resources (see Table 1). Moreover, our pattern also intends to support intrusion detection in cases where an intruder has circumvented authentication or other preventive safeguards.

First we consider the context where our pattern is applicable. We restrict our attention to systems providing services to an organisation (for instance, security or logistic services). Moreover, we consider systems of uniquely identifiable artificial entities, endowed with enhanced processing, communication and memory capabilities, such as robots or smart objects. Such capabilities enable entities to be aware of their history and to report on it. As a consequence, the system can exploit its knowledge of the history of the entities, and a history-based solution can be proposed for authentication and intrusion detection.

Several competing forces come into play in every authentication proposal, thus some of them must be optimised at the expense of others. On the one hand, the system must be protected against possible attacks, preserving the confidentiality of the relevant information and avoiding illegal manipulations. On the other hand, the available resources must be taken into account, so that the solution should assign resources proportionally to the criticality of the attack.

In the following subsections, we provide further details on the pattern and describe its structure, participants and behaviour. We conclude with an analysis of the proposal.

Table 1: History–based authentication pattern.

Name
History–based authentication
Context
A system of uniquely identifiable artificial entities, endowed with processing, communication and memory capabilities, such as robots or smart objects. The purpose of the system is to provide a service to an organization (e.g., security or logistic services).
Problem
For the sake of security, only valid entities should accomplish tasks related to the service. For this reason, the entities must be properly authenticated, that is, they must prove that they are what they claim to be. Moreover, situations in which an intruder entity has circumvented preventive safeguards such as authentication must be prevented by means of intrusion detection mechanisms.
Forces
- A third party could try to impersonate a legitimate entity. Different types of attack are possible, ranging from stealing identifying information to cloning the entity. The system should provide for prevention and early detection mechanisms. - Proportionality: the resources assigned to a task should be proportional to its frequency and criticality level. - Confidentiality: a third party eavesdropping on communications should not be able to obtain information that may be relevant for identification and authentication purposes. Thus communication of such information should be secure. - Tampering: the injection of incorrect data or the illegal manipulation of authentication-related data stored by an entity should be prevented or detected early.
Solution
The enhanced capabilities of the entities allow them to be aware of their history and process queries related to it. Taking advantage of this fact, authentication and intrusion detection can be based upon the entity's history. Some entities are specialised as verifiers. To exercise their role, they must be provided with adequate authority and capabilities, as well as knowledge about the history of the entities to be authenticated.

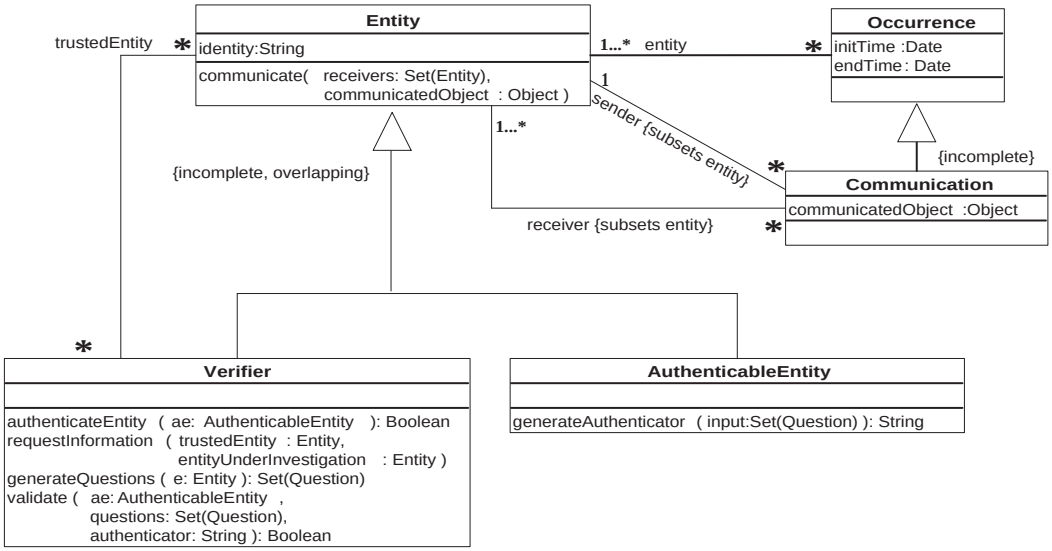

Figure 1: History–based Authentication Class Diagram.

4.1. Structure and Participants

As previously stated, we consider a system of uniquely identifiable entities endowed with enhanced capabilities. This fact has been represented in the UML class diagram of Figure 1 with the class Entity, which has an attribute (`Identity`) and several methods. We restrict ourselves to the methods that model the capabilities supporting our mechanism such as `communicate`, which represents the communication capability of entities.

We understand the history of an entity as a set of elementary components, which are in turn modelled by the notion of occurrence. An *occurrence* is defined as something (an event, an incident or something else) that happens to one or several entities during a period of time. We are aware that there is an even simpler concept defined as something that happens, without referring to any entity. This simpler concept is similar to the notion of event defined in the glossary (Luckham and Schulte, 2011) as "anything that happens, or is contemplated as happening". However, because our goal is to manage the entities' history, our atomic concept will be *occurrence*. It should be noted that our current concept of *occurrence* generalises the one we introduced in (Domínguez et al., 2013).

The association between the classes `Entity` and `Occurrence` models the ability of entities to remember occurrences regarding both their own history and that of other entities.

An entity builds up its history by registering the occurrences in which it is involved. The integrity of the history relies on meeting the following requirements every time an occurrence is recorded:

- The record reflects something that actually happened.
- The record reflects faithfully what really happened, that is, all of the recorded aspects are in accordance with the aspects of what happened.
- The record is complete, that is, the set of the recorded aspects allows us to determine and distinguish the occurrence that has been observed and that has been recorded.
- The record has not been tampered with, that is, once recorded, it has suffered no modification, perturbation or disguise.

Specialised types of occurrences can be derived from the basic notion. A type especially relevant to our proposal is *Communication*, which models the registration of a communication act. The registered occurrence is "an entity has taken part in an act of communication whose content is the communicatedObject". The association between `Occurrence` and `Entity` is specialised into two associations indicating the role that each entity plays (sender or receiver).

In particular, an entity can communicate with another one of its occurrences so that, in this case, the communicated object is an occurrence. As a consequence, an entity can acquire (at least a partial) knowledge of the history of other entities; this feature is highly relevant for our authentication mechanism.

Two different types of entities can be differentiated in the authentication process, the entity that has to be authenticated (authenticable entity) and the entity that authenticates it (verifier).

An *authenticable entity* is an entity capable of recording, recognising and reporting on its own history. In particular, it should be able to generate an output in response to an input consisting of any combination of questions regarding its history. This output, hereinafter referred to as the *authenticator*, proves that the entity knows the correct answers and serves to authenticate it to the system. The simplest case of an authenticator is just the concatenation of the answers to the posed questions. Authenticators may be strengthened by further processing the answers through computation methods such

as hashing functions, encryption with secret keys or zero knowledge proof techniques. We remark that an authenticable entity is inextricably linked to its history: if dissociated from it, the entity is no longer able to generate an authenticator and thus ceases being an authenticable entity. An example of a history-aware entity would be a smart thing such as a spime (Sterling, 2005), defined as a new category of space-time objects that are aware of their surroundings and can memorise real-world events; if provided with the capability to generate authenticators, a spime would be an authenticable entity.

On the other hand, a *verifier* is an entity endowed with both the necessary capabilities and the authority to authenticate other entities. To exercise its role, it must have access to the history of the entity to be authenticated. Such knowledge may be direct (derived from its own history) or communicated by other trusted entities.

We note that specialisation of `Entity` into `AuthenticableEntity` and `Verifier` is incomplete and overlapping. Incompleteness permits the existence of other sorts of entities in the system, such as sensors. In addition, overlapping allows a verifier to be, in turn, authenticated by another entity.

4.2. Dynamics

First, the authenticable entity AE identifies itself to the verifier V by means of the method `communicate(`V`, identity)`. The steps of the authentication process are described in Table 2, and shown graphically in Figure 2.

To authenticate AE, V first must possess sufficient knowledge about the history of AE. V can rely on its current knowledge or gather information from trusted entities (i.e., demand information on AE to a trusted entity TE through `requestInformation(`TE`,` AE`)` and receive a set of occurrences regarding AE through `communicate(`V, Set(O))). Depending on the centralised/distributed architecture of the system, such trusted entities could be a central repository, a set of credential service providers (whose credentials would authoritatively bind the identity of an entity and a subset of its history), or just trusted peers. Based upon the collected knowledge, V generates a set of questions Set(Q) (invoking `generateQuestions(`AE`)`)that are posed to AE (`communicate(`AE`,` Set(Q)), which in turn generates an authenticator (`generateAuthenticator(`Set(Q)`)`)and returns it to V (`communicate(`V`, authenticator)`). After validating the received authenticator (`validate(`AE`,`Set(Q)`, authenticator)`), V may reach a definitive conclusion on the authenticity

Table 2: Use case: AuthenticateEntity.

Identifier	AuthenticateEntity
Description	An entity is authenticated.
Involved actors	An authenticable entity, AE, and a verifier, V.
Main flow	1. AE identifies itself to V. 2. If necessary, V collects information about the history of AE from other entities in which V trusts (asks them to communicate occurrences regarding AE). Step 2 can be repeated as many times as considered necessary for V to have a sufficiently deep knowledge of AE's history. 3. V develops a battery of questions $(Q_1 \cdots Q_t)$ on the basis of its knowledge of AE's history. 4. V sends the questions to AE. 5. AE generates an authenticator to prove it knows the answers to the questions posed by V. 6. AE sends the authenticator to V. 7. V validates the authenticator received from AE. Steps 2 to 7 can be repeated as many times as necessary for V to obtain an acceptable result for the authentication of AE. 8. V communicates the result of the authentication process to AE.
Postconditions	AE is authenticated.
Alternative flows	If AE fails to prove its knowledge of the history of the entity it claims to be, the authentication is negative.

of AE's identity or pose new questions, possibly after gathering further information from its trusted parties.

To meet the proportionality requirement, the thoroughness of the questions posed during the interrogation depends on the criticality of the information/services/location the entity intends to access and the cause that triggered the authentication process. Some of the new questions can be made depending upon the answers formerly provided by AE (e.g., questions can be raised to clarify an ambiguous answer or to obtain more detailed answers).

Finally, V reports on the success or failure of the authentication process to AE (`communicate(`AE`, result)`).

4.3. Analysis of our proposal

We discuss the strengths and weaknesses of our history-based authentication proposal with respect to several properties. The list of properties to be analysed has been taken from (Kienzle et al., 2002).

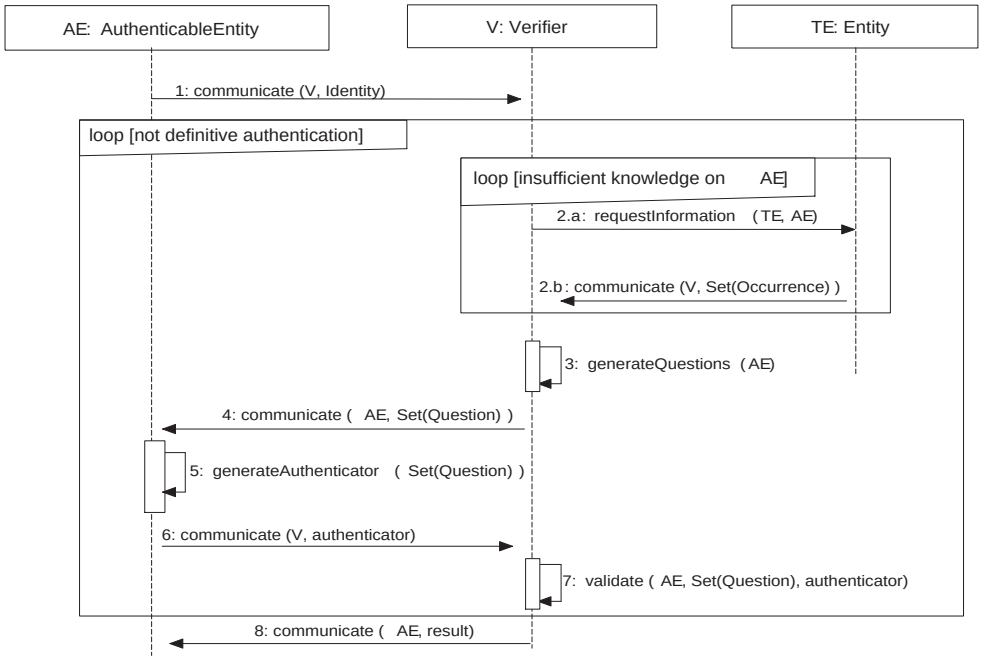

Figure 2: History–based Authentication Sequence diagram.

- *Accountability*: As well as any other type of authentication, our proposal permits individual entities to be held accountable for their actions. Furthermore, the more complex the authentication process is, the more difficult identity theft becomes. For this reason, attempts by entities to repudiate the transactions they initiate will be made more difficult.

- *Availability*: Our proposal could have an adverse effect on availability, resulting in denying access to legitimate entities. For example, an entity could forget its history, due to some type of hardware or software problem, making a required resource unavailable to it. On the other hand, our proposal enables the early detection or issues with an entity, allowing replacement or repair before the damage goes further.

- *Confidentiality*: Our proposal allows entity data to be protected from unauthorised disclosure with a greater level of confidence than other, simpler authentication proposals. An impersonator would need to steal the entire history of an entity to compromise confidentiality. Such an attack requires a much stronger effort than just stealing a password.

- *Integrity*: Like confidentiality, our proposal greatly enhances the integrity of data by enabling a stricter control of the entities involved in the storage, process and transit of information.

- *Manageability*: Due to the overhead required to maintain the history of the entities, using history has a larger impact on manageability than other, simpler authentication technologies, such as passwords. For this reason, our approach would be recommended when security is more important than manageability. Being aware of this issue, we establish the requirement of proportionality: the complexity of each realisation of authentication must be proportional to the criticality of the circumstances.

- *Usability*: This property is only applicable to human users. There must be a trade-off between the benefit of greater protection provided by stronger authentication procedures and the user's perception of extra security controls as a nuisance (Lee et al., 2012). Few people would be willing to answer demanding interrogations about his/her past to perform his/her day-to-day activities. Our model provides theoretical support for the design and implementation of devices that, once assigned to a human user, could trace his/her history and automatically provide an authenticator on his/her behalf whenever required. Security would be reinforced at no extra cost in terms of ease of use for the person possessing the device.

- *Performance*: Performance is affected by the need to store and manage the history of the entities. In general, choosing more secure procedures will generally have an adverse effect on performance.

- *Cost*: Costs will be incurred from the additional processing power required and the additional management overhead. However, this investment could cost less in the long term, because the system will be less vulnerable to attacks, which are inherently expensive to detect and recover from.

Taking into account the particular features of the previously described pattern, we now describe the main benefits of our proposal:

- It is fully applicable to artificial entities, provided they have sufficient storage and computational capacities.

- It allows dynamism and flexibility: the verifier can choose in any moment which and how many questions an entity must answer to be authenticated. The number and nature of questions can be adapted to the criticality of the resource an entity intends to access or to its observed behaviour. If an intrusion detection system raises suspicion about an already authenticated entity, a more exhaustive authentication process may be triggered.

- It allows collaboration: a verifier can complement its knowledge on the entity's history by consulting other trusted 'colleagues'. Such a feature is highly relevant in decentralised environments such as the IoT (Miorandi et al., 2012).

- It provides a basis for tampering detection: a thorough analysis on an entity's data may reveal inconsistencies (either internal or with the information stored in the entities belonging to the system) that may be an indicator of integrity loss.

- Moreover, it enables intrusion detection: the system can reconstruct the history of an entity from the memories of the components of the system that have interacted with it. Discrepancy between the historic appearance and behaviour and those currently perceived may be an indication that an intruder has entered the system.

5. Case study

In this section, we present a case study consisting of a security system based upon the previously described pattern. It has been designed within the framework of the project THOFU, a multidisciplinary collaboration that intends to perform a prospective study on technologic concepts and solutions enabling advanced services in the context of the hotels of the future. The project is being developed by a consortium composed of large companies, small and medium businesses and technology centres, funded by CDTI (Spanish Centre for Technological and Industrial Development) and supported by the Spanish Ministry of Science and Innovation (Consortium, 2010).

One of the main research areas of THOFU is related to security. Predictably, monitoring of comfort, safety and security of hotels and many other social and industrial environments of the future will be assigned to pervasive systems. Such systems, operating with little or no human intervention,

should in turn control and protect themselves to assure their correct performance and their compliance with requirements such as privacy and security.

In this section, we outline the security system developed within the THOFU framework, the THOFU Security System, hereinafter referred to as TSS. We focus on the features of TSS related to history awareness, either as beneficiaries or as enablers and supporters.

5.1. Description

The system is composed of entities that can be categorised into several types, according to their roles:

- **robots**, responsible for the surveillance of the facilities and resources of the organisation. Because they are required to identify and authenticate themselves to the system, robots are a type of the authenticable entities described in Section 4.1.

- **controllers**, responsible for monitoring the performance of other entities (e.g., robots and other controllers). In case an asset (resource or capability) is unavailable or underperforms, they trigger a corrective action. Because they should be able to authenticate the entities under their control, controllers are a type of the verifiers described in Section 4.1. Moreover, controllers which are supposed to be in turn controlled by other entities are also *authenticable entities*.

- **registrar**, which establishes the identity of every new entity that is incorporated into the system. Moreover, it assures that both the new entity and the system possess sufficient information and capabilities to perform the authentication process.

- **knowledge manager**, a subsystem that registers the history of the system as a whole, supporting the knowledge management processes. Such knowledge could be observed as a very preliminary stage of the system's conscience.

5.2. Security requirements

To adequately face both the forces of our pattern (see Table 1) and those naturally arising during the development of TSS, a set of security requirements is established. They should be met during the service life of an authenticable entity AE (e.g., a robot or a controller).

R1 The system must not possess all the information AE uses to authenticate itself. In this way, an attacker with access to all of the information about AE available in the system lacks vital information to impersonate it. This requirement generalises standard requirements regarding password management. See, for example, security goals G1 (No verification table), G3 (No password reveal) and Security Requirement SR9 (Prevention of insider attacks) in (Madhusudhan and Mittal, 2012).

R2 History secrecy: AE leaks no useful information about its occurrences to non-authorised parties such as eavesdroppers.

R3 Anonymity: a third party that eavesdrops on the communications between AE and other entities in the system must not be able to identify AE with a reasonable effort. This is one of the goals (G8: User anonymity) raised in (Madhusudhan and Mittal, 2012).

R4 Unlikability: an eavesdropper should not be able to trace the history of the interactions between AE and the system. In particular, the link between AE and its communications (individuating the communications where AE has taken part) should not be established with a reasonable effort.

R5 Unforgeability: only AE should be able to perform valid communications on its behalf.

R6 Proportionality: the amount of resources required for interactions between AE and the system (in particular, their computational cost) is proportional to the frequency and criticality level of the interaction.

R7 Intrusion detection: if a third party succeeds in impersonating a legitimate entity, it should be unmasked as soon as possible.

R8 Fault and Tampering detection: the loss of the integrity of the information stored in an entity (due to data corruption or malicious tampering) should be detected as soon as possible.

5.3. An enabler for authenticator generation

As described in Section 4, authentication in our proposal relies on the common knowledge of (at least part of) the history of AE, $\mathrm{Set(O)_{AE}}$, which can be considered as a secret shared between AE and the system.

However, such common knowledge is not enough to assure the security requirements posed in Section 5.2. Requirement R1 suggests that AE should possess a secret key, SK_{AE}. For the sake of security, AE must not communicate the value of SK_{AE} but prove its knowledge (using, e.g., zero knowledge proof techniques; see (Camenisch and Stadler, 1997)).

On the other hand, requirement R2 in 5.2 implies the use of a procedure to encode messages interchanged between authorised parties within the system. We suggest the use of a hash function $H : \{0,1\}^* \rightarrow \{0,1\}^k$, with k being a sufficiently long integer. Upon registration, the system provides AE with the capability to compute the function H, either simply by communicating the algorithm for its computation or by equipping AE with a hardware device that encodes H. In this last case, authentication would be multifactor (the entity knows SK_{AE} and $\mathrm{Set(O)_{AE}}$ and has H), and thus, the security would be enhanced. Moreover, protecting the device with a physical security mechanism, as advised in (Burr et al., 2011), would provide a safeguard against tampering and duplication attacks.

In our proposed pattern, an authenticable entity must be able to generate an authenticator. To implement such capability, AE uses the tuple $(SK_{AE}, \mathrm{Set(O)_{AE}}, \mathrm{H})$ (see Section 5.5.3 and Table 5). This tuple acts as a token, i.e., something possessed and controlled by an entity, which is used to authenticate its identity (Burr et al., 2011). Note that, because a history grows over time, the token gets more and more complex (in size and complexity) and thus more and more difficult to steal or forge. The strength of the token is thus an increasing function of time.

5.4. Lifecycle

In this section, we will sketch the lifecycle of an authenticable entity AE (a robot or a controller) within the security system TSS.

The goals of the description are threefold. First, we intend to illustrate the compliance of the policy stating that the system, TSS, is responsible for assuring its own security and monitoring its own performance. In particular, during its operating phases, AE will be under the control of another component of the system that assures an early reaction in case of misbehaviour or underperformance. Second, our purpose is to show our proposal at work, by focusing on the processes that benefit from the history-based mechanism and on those processes that support it. Finally, our description should be thorough enough to account for meeting the requirements in Section 5.2.

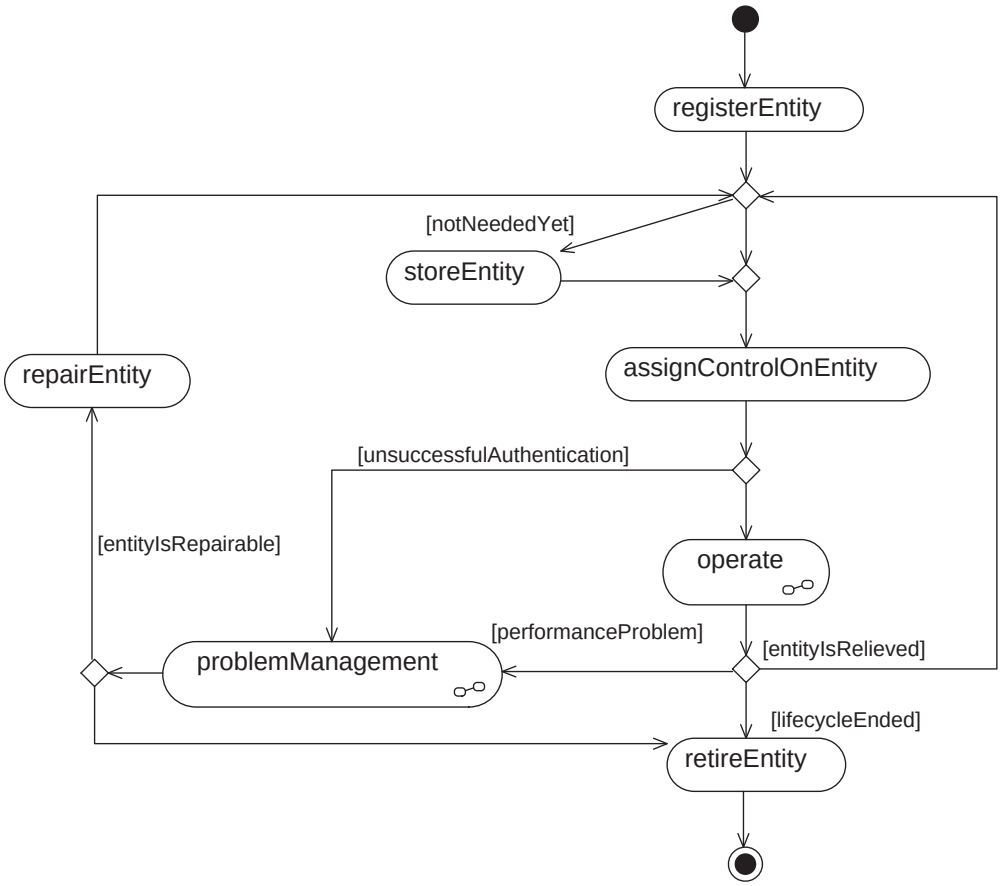

Figure 3: Lifecycle.

First, AE must be registered within the system (see Figure 3). In the registration phase, the registrar establishes AE's identity after validating the identity proofs provided by AE (e.g., a credential issued by a trusted system from where AE comes). The registrar also receives a commitment of AE's secret key $f(SK_{AE})$, together with an initial set of occurrences from its previous history, $\mathrm{Set(O)_{AE}}$. The purpose of $f(SK_{AE})$ is enabling AE to construct Zero Knowledge Proofs of Knowledge of SK_{AE} (AE must be able to prove it knows SK_{AE} without unveiling its value). If AE comes straight from the factory, the initial set of occurrences may relate to its provenance (e.g., date of manufacture and fabricant). The registrar provides AE with the capability to compute a hash function H, either by communicating the code of H to AE or by equipping AE with a hardware device with H encoded. For the sake of simplicity, H is supposed to be common for all of the entities within the system. Now AE possesses a preliminary token $(SK_{AE}, \mathrm{Set(O)_{AE}}, \mathrm{H})$, and the registrar has enough information to authenticate AE.

Once registered, AE may be either assigned a task or stored while waiting for an assignment. Whenever AE is assigned a task, its control is in turn assigned to a verifier V that will be a controller. To endow V with the capability to authenticate AE, the knowledge manager signs a credential binding the identity of AE to its token, $sign(Id_{AE} \| f(SK_{AE}) \| \mathrm{Set(O)_{AE}})$, and sends it to V. For the sake of security, the credential can be encrypted using V's public key.

To meet requirement R3 (anonymity) AE computes a temporary identifier, Id'_{AE}, upon receiving instructions from V. To meet requirement R4 (unlinkability), AE changes Id'_{AE} from time to time at the behest of V. For details on generating Id'_{AE}, see use case UpdateId' below, Section 5.5.1.

During the operation phase, AE and V communicate on a continuous basis. Different communication types, depending on their purpose, take place:

- AE maintains contact with V by sending signals to ensure that V knows it is active. Because this type of communication occurs frequently, to meet requirement R6 (proportionality), the signal must be easy to generate by AE and to process by V. A possible signal fulfilling this condition is just the value Id'_{AE}.

- AE communicates occurrences related to the performance of its assigned task to V (see use case CommunicateOccurrenceToController below, Section 5.5.2).

In our proposal, building a new temporary identifier or communicating a new occurrence requires an up–to–date knowledge of the history of AE. Failing to perform such activities is a sign that there is something wrong with AE: inability to account for its own history may be due to a communication error (either in the reception of the request made by V or in the transmission of the message), memory corruption (AE has forgotten part of its history) or the entity being an impostor impersonating AE. Thus, our proposal provides support for intrusion detection and fault and tampering detection (requirements R7 and R8). V reacts to this situation by demanding that AE authenticate itself. If AE succeeds and no errors arise in the following communications, V may decide to close down the incident. Otherwise, AE is put into quarantine and V transfers the problem to a specialised subsystem that performs in–depth hardware and software testing on AE, decides the subsequent step (e.g., repair or retire AE) and generates an error record.

There are different circumstances under which V may demand authentication from AE:

- Prior to assuming control of AE.
- On a random basis, without the need for a specific cause. The random checks enhance the security (e.g., to unmask an impersonator who has eavesdropped on AE's identifying signal and uses it to impersonate AE).
- After a silence time that is too long: V does not receive messages from AE within a stipulated security time interval. This silence may be due to acceptable causes (there is a natural physical obstacle between AE and V interfering with communications), the malfunction of AE, malicious interferences or an attack that has eliminated AE.
- In general, as a response to AE underperforming.

Authentication is performed under the guidelines of our pattern (Subsection 4.2). An adaptation of the use case AuthenticateEntity described in Table 2, taking into account the features of TSS, will be provided below (see use case AuthenticateEntity_TSS below, Section 5.5.3).

The operation phase finishes in any of the following cases:

- AE successfully completes its task.

- AE is relieved by another entity because new capabilities are required for performing the task, AE has been assigned a more urgent task or AE has worn out or become obsolete.

- Problems with the performance of AE, as noted above.

In all cases, when the operation phase is complete, V proceeds to update the knowledge the system has about the history of AE.

5.5. Use cases

In this section, the use cases *UpdateId'*, *CommunicateOccurrenceToController* and *AuthenticateEntity_TSS*, previously presented, are described. They exploit the knowledge V has about the history of AE.

5.5.1. UpdateId'

The purpose of this use case is to change the temporary identifier AE uses to maintain contact with V, Id'_{AE}, at the behest of V.

The use case proceeds as follows. First, V sends AE several questions related to AE's history. Then, AE computes its new temporary identifier, Id'_{AE}, applying the hash function H to the concatenation of its identifier Id_{AE} and the answers to the posed questions. A detailed description of the use case is provided in Table 3.

This use case enables the system to meet requirements R3 (anonymity) and R4 (unlinkability). It also enables intrusion and fault and tampering detection processes (requirements R7 and R8), as explained below:

An eavesdropper without knowledge on AE's history cannot deduce Id_{AE} from its pseudonym Id'_{AE}. Let us suppose, on the other hand, that it has been eavesdropping on the system for a long period of time and has thus registered the temporary identifiers used by the different authenticable entities. It cannot relate each entity AE to the set of its successive identifiers, and, thus, cannot reconstruct the history of its interactions with the system.

A third party A' that has succeeded in impersonating AE but lacks a thorough knowledge of its history can maintain the hoax only until V asks it to update its temporary identifier. Its incapability to build a new identifier recognisable by V would give it away, and its intrusion would be detected.

On the other hand, if an entity suffers memory corruption due to natural causes or an attack, or if fake information has been injected into it, it would fail when performing the use case UpdateId'. As this use case is supposed to be executed frequently, the problem would be detected at an early stage.

Table 3: Use case: UpdateId'.

Identifier	UpdateId'
Description	A new value is assigned to Id'_{AE}, the temporary identifier of AE.
Involved actors	Entities AE and V (V controls AE).
Preconditions	Both entities are registered in the system. Monitoring of AE has been assigned to V; thus, V knows $(Id_{AE}, f(SK_{AE}), \mathrm{Set(O)_{AE}})$
Main flow	1. Within a maximum interval Δt_{Update}, V sends AE a tuple of questions $(Q_1 \cdots Q_t)$ related to the history of AE, whose answers are known by V. V can sign the tuple to assure he is really the sender of the message. 2. AE processes the questions and obtains the answers $(A_1 \cdots A_t)$. 3. AE computes $Id'_{AE} = H(Id_{AE} \| A_1 \| \cdots \| A_t)$, where H and $\|$ stand for a hash function known by AE and V and the concatenation operator.
Postconditions	A new temporary identifier for AE, Id'_{AE}, has been obtained.
Alternative flows	

5.5.2. CommunicateOccurrenceToController

To communicate an occurrence o to V, AE generates a coded message m, which is a function of o and a subset of the history of AE shared with V. In particular, m is computed by applying the bitwise AND operator to the occurrence to be communicated and the answers to the last battery of questions posed to AE by V (see Table 4). Then, AE sends V its temporary identifier, Id'_{AE}, and the message m. Finally, V identifies AE by means of Id'_{AE} and extracts o from m.

A third party without such knowledge of AE's history can neither recover o from m nor build a valid message on behalf of AE. Thus, requirements R2 (history secrecy) and R5 (unforgeability) are met.

5.5.3. AuthenticateEntity_TSS

AuthenticateEntity_TSS adapts the use case *AuthenticateEntity* (see Table 2) to our test bed system. The main adaptations consist of adding the specific manner in which the identification and the authenticator generation are performed to the use case. On the one hand, AE identifies itself to V by means of its temporary identifier. On the other hand, we detail the manner in which AE generates an authenticator: AE computes a message m by

Table 4: Use case: CommunicateOccurrenceToController.

Identifier	CommunicateOccurrenceToController
Description	AE communicates an occurrence to V.
Involved actors	Entities AE and V (V controls AE).
Preconditions	Both entities are registered in the system. Monitoring of AE has been assigned to V; thus, V knows $(Id_{AE}, f(SK_{AE}), \mathrm{Set(O)_{AE}})$
Main flow	1. AE processes $m = o*(A_1 * \cdots * A_t)$, where o is the occurrence to be communicated, expressed in a given language, $(A_1 \cdots A_t)$ are the answers to the last battery of questions posed by V to AE, and $*$ is the bitwise AND operator. Note that A_i relate to previous occurrences stored in the history of AE, known by both AE and V, while o is a new occurrence that AE wants to communicate to V. 2. AE sends V the tuple (Id'_{AE}, m), where Id'_{AE} is the temporary identifier of AE. 3. V receives the message. Through Id'_{AE}, V identifies AE. 4. V processes $o = (A_1 * \cdots * A_t) * m$ and thus reconstructs o.
Postconditions	AE has communicated the occurrence o to V.
Alternative flows	V receives an invalid communication, because the result of $(A_1 * \cdots * A_t) * m$ contains syntax errors or is inconsistent (i.e., does not comply with the syntax of the language used for communicating occurrences, or its semantics is nonsense). In this case, V proceeds to check AE's identity by launching an authentication process.

concatenating the answers to the questions posed by V ($m = A_1||\cdots||A_t$) and generates a zero knowledge proof of knowledge of its secret key SK_{AE} on the message m: $proof[SK_{AE}](m)$. To do so, techniques such as those proposed by (Camenisch and Stadler, 1997) can be used. (An example of how to build such proof is provided in Appendix A). The complete use case appears in Table 5. To ease the reading of the use case, additions are marked in bold.

We note that this use case meets requirement R2 (history secrecy): the authenticator generated by AE leaks no information about the history of AE. On the other hand, a third party unaware of SK_{AE} or the history of AE cannot successfully authenticate itself, meeting requirement R5 (unforgeability). This use case provides sufficient flexibility to allow meeting requirement R6 (proportionality): the thoroughness of the interrogation suffered by AE may vary, depending on the cause that triggered the authentication process.

5.6. Meeting Requirements

In this section we summarise how our system satisfies the requirements stated in subsection 5.2:

- R1 is satisfied because AE communicates no information that may be useful to rebuild its secret key SK_{AE}, neither to the registrar nor to its successive controllers. AE only provides Zero Knowledge Proofs of the knowledge of SK_{AE}.

- R2 (History secrecy) and R5 (Unforgeability) are met by both *CommunicateOccurrenceToController* and *authenticateEntity*.

- R3 (Anonymity) and R4 (Unlinkability) are enabled by use case *UpdateId'*.

- R6 (Proportionality): the most frequent communications between AE and V (sending Id'_{AE} to maintain contact) requires minimal processing by V. In addition, our proposal provides an authentication process with flexibility to adapt its thoroughness to the context.

- R7 and R8 (Intrusion and fault & tampering detection) are reinforced by the use case *UpdateId'*, which allows an early detection of such threats.

Table 5: Use case: AuthenticateEntity_TSS.

Identifier	AuthenticateEntity_TSS
Description	An entity, **formerly registered by the system**, is authenticated.
Involved actors	An authenticable entity, AE, and a verifier V .
Preconditions	AE has been previously registered in the system. **Monitoring of AE has been assigned to V; thus, V knows** $(Id_{AE}, f(SK_{AE}), \mathrm{Set(O)_{AE}})$
Main flow	1. AE identifies itself to V **through its temporary identifier,** Id'_{AE}. 2. If necessary, V collects information about the history of AE from other entities in which V trusts (asks them to communicate occurrences regarding AE). 3. Step 2 can be repeated as many times as considered necessary for V to have a sufficiently deep knowledge of AE's history. 4. V elaborates a battery of questions $(Q_1 \cdots Q_t)$ based on its knowledge of AE's history. 5. V sends the questions to AE. 6. AE **computes a message m by concatenating the answers to the questions posed by V, $m = A_1 \| \cdots \| A_t$, issues a zero knowledge proof of knowledge of its secret key on message m,** $proof[SK_{AE}](m)$ and sends the result to V. 7. V validates the authenticator **proof** $[SK_{AE}](A_1 \| \cdots \| A_t)$ received from AE. 8. Steps 2 to 7 can be repeated as many times as necessary for V to obtain an acceptable result for the authentication of AE. 9. V communicates the result of the authentication process to AE.
Postconditions	AE is authenticated.
Alternative flows	If AE fails to prove knowledge of SK_{AE}, or of the history of the entity it claims to be, the authentication is negative.

6. Conclusions and future work

A history-based mechanism has been proposed to reinforce the authentication processes, either by itself or when complementing other approaches. It also supports intrusion detection.

Our proposal relies on the basic notion of *occurrence*, which supports the notions of *history* and *authenticable entity*. It can be applied to systems with entities endowed with communication, memory and processing capabilities. Such capabilities enable the implementation of the previous notions.

Among the advantages of our proposal, we highlight its dynamism and flexibility. Moreover, it enables collaboration and supports tampering and intrusion detection. As a result, it helps to prevent unauthorised disclosure of data and malicious information modification or destruction, and to assure the identity of the entity that has performed an action. Thus, confidentiality, integrity, authenticity and non-repudiation are strengthened (ISO, 2012), (Ross et al., 2008).

There are several lines of further work. Several aspects of occurrence management are left open, such as persistence structures for storing the occurrences and a model for communicating occurrences (including its syntax and semantics). In addition, a goal for further work is to analyse the utility and feasibility of our history-based approach for implementing other features of the IoT (Miorandi et al., 2012) apart from authentication.

Acknowledgements

This work has been partially supported by the Spanish Ministry of Science and Innovation (projects TIN2009- 13584 and SMOTY, IPT-2011-1328-390000), the Centre for Industrial Technological Development (CDTI) (project THOFU, CEN20101019), the Government of Aragón and the European Social Fund.

Appendix A. Zero Knowledge Proof of Knowledge of SK_{AE}

In this section, we suggest a possible algorithm providing a zero knowledge proof of knowledge of SK_{AE} and secure transmission of questions and answers that could be used in the *AuthenticateEntity_TSS* use case (Section 5.5.3). We follow the techniques of (Camenisch and Stadler, 1997) relating to zero knowledge proofs.

Notations

Given a cyclic group $G =< g >$, the discrete logarithm of $y \in G$ to the base g is the smallest integer r so that $g^r = y$.

$\|$ denotes the concatenation of two strings (possibly representing integer numbers)

$\mathbb{Z}_n^*$ stands for the multiplicative group of integers modulo n.

Initialisation

AE computes:

- An RSA public key (n, e), together with its private key, d, (Rivest et al., 1978).
- A cyclic group $G =< g >$ of order n in which computing discrete logarithms is infeasible.
- A secret key taken at random $SK_{AE} \in \mathbb{Z}_n^*$.
- The element $y = g^{SK_{AE}}$.

Registration

In the registration phase, AE communicates initial information to the registrar: its identifier Id_{AE} and its public key (n, e, G, g, y), together with information on its provenance. Note that this public key replaces $f(SK_{AE})$, as denoted in the description of the registration phase (Section 5.4).

When V is charged with the monitoring of AE, it receives both Id_{AE} and the public key.

An authentication algorithm

1. V develops a battery of questions related to the history of AE known by V, $(Q_1 \cdots Q_t)$.
2. V encrypts them (e.g., by using AE's public key, n, e) and sends the resulting $(Q_1^e \cdots Q_t^e)$, where exponentiations are performed modulo n.
3. AE decrypts the questions: $Q_i = (Q_i^e)^d$ modulo n.
4. AE obtains the answers $(A_1 \cdots A_t)$ and concatenates them: $m = A_1 \| \cdots \| A_t$.
5. AE computes (c, s), where $c = H(m \| y \| g \| g^r)$ for a random $r \in \mathbb{Z}_n^*$, and $s = r - c \cdot SK_{AE}$.

6. AE sends the pair (c, s) to V.

7. V verifies $c =?H(m\|y\|g\|g^s y^c)$, where $m = A_1\| \cdots \|A_t$. Note that V knows all the required values: (n, e, G, g, y) have been communicated when it has been assigned to monitor AE, and $(A_1 \cdots A_t)$ are the answers to the questions it has posed to AE.

The ability to compute the pair (c, s) proves that AE knows the discrete logarithm of y to the base g (i.e., SK_{AE}). Thus (c, s) is an acceptable proof of knowledge of SK_{AE} on the message m. We note that m depends on the questions raised by V, which are different every time authentication occurs and unknown a priori by AE.

References

Adi W. Mechatronic security and robot authentication. International Journal of Advanced Science and Technology 2010;14:41–52.

Akyildiz IF, Su W, Sankarasubramaniam Y, Cayirci E. Wireless sensor networks: a survey. Computer networks 2002;38(4):393–422.

Atzori L, Iera A, Morabito G. The internet of things: A survey. COMPUTER NETWORKS 2010;54(15):2787–805.

Bhargav-Spantzel A, Squicciarini AC, Xue R, Bertino E. Multifactor identity verification using aggregated proof of knowledge. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on 2010;40(4):372–83.

Bicchi A, Fagiolini A, Pallottino L. Toward a society of robots behaviors, misbehaviors, and security. IEEE ROBOTICS & AUTOMATION MAGAZINE 2010;17(4):26–36.

Biermann E, Cloete E, Venter LM. A comparison of intrusion detection systems. Computers & Security 2001;20(8):676–83.

Burr WE, Dodson DF, Newton EM, Perlner RA, Polk WT, Gupta S, Nabbus EA. Electronic authentication guideline. NIST Special Publication 800-63-1 2011;Available at: http://csrc.nist.gov/publications/nistpubs/800-63-1/SP-800-63-1.pdf. Last visited on May 2013.

Caire P, van der Torre L. Convivial ambient technologies: Requirements, ontology and design. The Computer Journal 2010;53(8):1229–56.

Camenisch J, Stadler M. Efficient group signature schemes for large groups. In: Advances in CryptologyCRYPTO'97. Springer; 1997. p. 410–24.

Cobb WE, Laspe ED, Baldwin RO, Temple MA, Kim YC. Intrinsic physical-layer authentication of integrated circuits. Information Forensics and Security, IEEE Transactions on 2012;7(1):14–24.

Consortium T. Thofu: Technologies of the hotel of the future; 2010. Available at: http://www.thofu.es/seguridad. Last visited on June 2013.

Davis NZ. The Return of Martin Guerre. Harvard University Press, 1983.

Domínguez E, Pérez B, Rubio L, Zapata MA, Lavilla J, Allué A. Occurrence-oriented design strategy for intelligent and quality systems development; 2013. Submitted to IEEE Transactions on Knowledge and Data Engineering.

Gavrilova ML, Yampolskiy RV. State-of-the-art in robot authentication. IEEE ROBOTICS & AUTOMATION MAGAZINE 2010;17(4):23–4.

González Alonso I, Álvarez Fres O, Alonso Fernández A, del Torno PG, Maestre J, Almudena García Fuente M. Towards a new open communication standard between homes and service robots, the dhcompliant case. Robotics and Autonomous Systems 2012;60(6):889–900.

ISO . Information technology - security techniques - information security management systems - overview and vocabulary, iso/iec standard 27000:2012, 2012-12-01. 2012.

Jolly J. Helizabeth hawley, unrivalled himalayan record keeper; 2010. Available at: http://www.bbc.co.uk/news/world-south-asia-1026854. Last visited on May 2013.

Kienzle DM, Elder MC, Tyree D, Edwards-Hewitt J. Security patterns repository version 1.0. DARPA, Washington DC 2002;Available at: http://www.scrypt.net/ celer/securitypatterns/repository.pdf. Last visited on May 2013.

Lee JER, Rao S, Nass C, Forssell K, John JM. When do online shoppers appreciate security enhancement efforts? effects of financial risk and security level on evaluations of customer authentication. International Journal of Human-Computer Studies 2012;70(5):364–76.

Lee Brown F, Di Vietri J, Diaz de Villegas G, Fernandez E. The authenticator pattern. In: Conference on Pattern Languages of Programming (PLoP). 1999. .

Luckham D, Schulte R. Event processing glossary-version 2.0. Online Reference 2011;Available at: http://www.complexevents.com/2011/08/23/event-processing-glossary-version-2-0/. Last visited on May 2013.

Madhusudhan R, Mittal R. Dynamic id-based remote user password authentication schemes using smart cards: A review. Journal of Network and Computer Applications 2012;35(4):1235–48.

Miorandi D, Sicari S, Pellegrini FD, Chlamtac I. Internet of things: Vision, applications & research challenges. Ad Hoc Networks 2012;10(7):1497–516.

Ning H, Liu H. Cyber-physical-social based security architecture for future internet of things. Advanced in Internet of Things 2012;2(1):1–7.

Paci F, Ferrini R, Musci A, Steuer K, Bertino E. An interoperable approach to multifactor identity verification. Computer 2009;42(5):50–7.

Rivest RL, Shamir A, Adleman L. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM 1978;21(2):120–6.

Roman R, Najera P, Lopez J. Securing the internet of things. Computer 2011;44(9):51–8.

Ross R, Katzke S, Johnson A, Swanson M, Stoneburner G. Nist sp800-39, managing risk from information systems an organizational perspective. 2008.

Ruiz-Del-Solar J, Verschae R, Arenas M, Loncomilla P. Robot detection system in the soccer domain. IEEE robotics & automation magazine 2010;17(4):43–53.

Sterling B. Shaping things. MIT press, 2005.

Suh GE, Devadas S. Physical unclonable functions for device authentication and secret key generation. In: Proceedings of the 44th annual Design Automation Conference. ACM; 2007. p. 9–14.

Wang Y. Cognitive robots. a reference model toward intelligent authentication. Robotics & Automation Magazine, IEEE 2010;17(4):54–62.

Ye N, Emran SM, Chen Q, Vilbert S. Multivariate statistical analysis of audit trails for host-based intrusion detection. Computers, IEEE Transactions on 2002;51(7):810–20.