Security in Virtual Worlds, 3D Webs, and Immersive Environments:

Models for Development, Interaction, and Management

Alan Rea
Western Michigan University, USA

Information Science REFERENCE

INFORMATION SCIENCE REFERENCE

Hershey · New York

Director of Editorial Content: Kristin Klinger
Director of Book Publications: Julia Mosemann
Acquisitions Editor: Lindsay Johnston
Development Editor: Joel Gamon
Publishing Assistant: Julia Mosemann, Natalie Pronio
Typesetter: Natalie Pronio
Production Editor: Jamie Snavely
Cover Design: Lisa Tosheff

Published in the United States of America by
Information Science Reference (an imprint of IGI Global)
701 E. Chocolate Avenue
Hershey PA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@igi-global.com
Web site: http://www.igi-global.com

Library of Congress Cataloging-in-Publication Data

Security in virtual worlds, 3D webs, and immersive environments : models for development, interaction and management / Alan Rea, editor.
p. cm.
Includes bibliographical references and index.
Summary: "This publication discusses the uses and potential of virtual technologies and examines secure policy formation and practices that can be applied specifically to each"--Provided by publisher.
ISBN 978-1-61520-891-3 (hardcover) -- ISBN 978-1-61520-892-0 (ebook) 1. Computer networks--Security measures. 2. Web sites--Security measures. 3. World Wide Web--Security measures. I. Rea, Alan.
TK5105.59.S442 2011
005.8--dc22
2010045520

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

Chapter 9
Property-Based Object Management and Security

Torsten Reiners
Curtin University of Technology, Australia & University of Hamburg, Germany

Sascha Wriedt
University of Hamburg, Germany

Alan Rea
Western Michigan University, USA

ABSTRACT

The hype of Second Life is over. But the experience of this truly exciting period lives on in many disciplines and research areas, which are developing emerging technologies in virtual, as well as augmented worlds. And as is the rule with new forming developments, the path is not yet determined and weaves through different stages and platforms, calling for additional prototypes to understand the true impact of virtual worlds, Web 3D, or Augmented Reality. Using broad strokes and looking for a common denominator, most people conclude that it is Web 2.0 with all its (social) functionality and 3D objects as the embodiment of virtual existence. Many publications discuss Web 2.0 features and applications, but most do not focus on the 3D objects in the context of virtual worlds and their implications. In this chapter, the authors examine and observe what (virtual) objects are, as well as which properties should be used for inter-world interoperability. The past technological implementations demonstrate that protecting digital media (i.e. music and video) is an endless endeavor and that no security feature is simultaneously unbreakable and usable. This does not need to be the case for 3D virtual objects because we can learn from the past and achieve a new level of protection in a rising media. In this chapter the authors propose such a solution by putting forth a general 3D object understanding that includes a look at virtual worlds such as Second Life with a feasible concept of object security. They suggest that with a new framework objects can be secured and promote additional growth within, and among, virtual worlds. They propose a Global Object Management System (GOMS) architecture as a potential solution to this challenge.

DOI: 10.4018/978-1-61520-891-3.ch009

1 INTRODUCTION

When we think about security in virtual worlds, our first inclination might be how we login into the system and verify our identity. This single-factor authentication by username and password is meant to protect our identity, and privileges in the world. However, we are also protecting the true value of the virtual world avatar: its owned objects. Of course we can consider the experiences through play (Castronova, 2003) as well, but what we might value the most is not just the avatar (and by extension, us) but owned objects (e.g., property) (Horowitz, 2007). Of course these objects can be placed in the virtual world, worn on the avatar, or stored in a virtual inventory, but they generally exist in a single virtual world. Depending on the world, users might have invested massive amounts of real or virtual money, or copious amounts of time to upgrade items. Because of these investments, securing virtual objects within specific worlds, and potentially transferring them to others (in the case of virtual world closure, user choice, etc.) is quite an important issue now and in the future as we see greater virtual market penetration via 3D environments.

Managing objects is already a critical need even with the majority of current virtual worlds as closed environments. In closed environments avatars and objects are restricted to just one environment and a direct transfer is seldom possible or even favored. Even though the object encoding standard for the graphical representation of objects (nodes, edges, textures) is shared, additional scripting or environment specific modifications cannot be exported. Virtual worlds, such as World of Warcraft (Blizzard Entertainment, 2010) or Habbo Hotel (Habbo Hotel, 2010), do not allow object imports to protect the in-world market. In these worlds, providers place the objects in the virtual environment's economy, and offer them in specialized stores or reward users after they complete a series of tasks (e.g. quests). There are some trades completed in marketplaces like eBay, but many times these violate the specific virtual world's Terms of Service (TOS). Within a virtual game realm, there are reasons for this control, such as the provider needing to control the game flow, balance, and economy. Virtual worlds are different. Development over the last few years indicates a stronger desire for object interchange between virtual worlds (Watte & Systems, 2009). In order to effectively facilitate this interchange, the interoperability of objects beyond the specific world's aspects and controls is crucial. In other words, users' objects need to retain privileges, functionality, and interaction with the chosen virtual world environment whenever possible. We propose that the replacement of the current paradigm of environmentally-controlled static objects with dynamic objects that communicate their properties, much as newer object oriented programming languages.

1.1 Discussion of Crucial Terminology

Before continuing, we need to define our terminology for this chapter as our use of interpretation of terms might be different. We use the term *object* for virtual (digital) objects in virtual worlds. Note that the analogy to the object model as used in programming languages like C++ or Java basically matches our concept of object representation: Objects behave according to their code, interact by messages, and are related by a dynamic reference graph with relations as part of a larger semantic network (cf. Section 2.2). Objects belonging to an avatar are stored in an individual *inventory* or general storage facilities (*repository)*. *Properties* specify the object; this is also called metadata (Baca, 2008) or capabilities (Scheffler et al., 2008). The creator, owner, or user of an object is represented in virtual worlds by *avatars*. Note that in our understanding, *users* on social networks like Facebook or Flickr are as well using (multiple) avatars to represent their *identity*. We do adhere to the term avatar for the virtual representation of

one user and ignore possible multiple identities (c.f. post-modern identity theory) depending on the current control. We use the term *users*, even though in virtual environments, this does not imply only humans but could stand for an object, which acts like or on the behalf of the real world person. Therefore, everything being said about the user is also valid for objects. Objects require access rights and privileges to interact in the environments. This is a critical distinction for us as we use the term *access* with respect to resources in the environment rather than access to the environment (*authentication; identity verification* of the real user*)*. We use the term *group* to describe a collection of users (and ultimately objects), whereas we use the term *entity* for user, group, or object.

1.2 Moving from Avatar to Object

Pushing the awareness away from the avatar-centered perspective in virtual worlds toward the object itself is the underlying theme throughout our discussion, as well as our proposed architecture. In every context, researchers talk about the experiences in virtual worlds (Gregory et al., 2009), how users represent themselves (Ducheneau et al., 2009), how communication is taken to a higher level (Reiners et al., 2009), and how students explore and even create their learning material (Erlenkötter et al., 2008). But the object, the true building block of every idea, plays just a minor supporting role in most research and discussions. Obviously, the creations are demonstrated in words and images, and used to emphasize the new learning experience, but rarely does research examine securing the creations, neither archiving nor transferring them to other platforms. Perhaps most researchers consider this the work of the hosting organization, but we would argue this is not the case.

Most users would agree that objects are critical for the immersive experience; otherwise, virtual worlds would be full of avatars with nothing to do (and some might argue avatars would not even exist). Users from around the world have already created over a billion Second Life objects, but these are objects stored on closed servers with access only by specialized software with limited options for local copies and transfer to other environments. This paradigm extends to many other worlds as well. We argue that although the concentration is on the avatar for identification, the value of objects is often underestimated. In this chapter, we explore the object itself and describe the minimal requirements for relevant properties in secure (multi-) environments. By examining the importance of objects and illustrating the need to control and manage every aspect, we are able to create the foundation on which we design our concept that defines how one can manage and secure not only the objects themselves but also increase the potential for the inter-world transfer of objects.

We must stress that although we accentuate the importance of identity security, authentication, and trusted networks, we do not discuss detailed technical aspects, but remain at a high-level discussion. Our lack of discussion in this area is not meant to slight its importance, nor does this mean that we do not address methods to secure objects like digital watermarking; however, we provide only references for algorithmic or implementation details as these are far beyond the scope of this chapter, but not our overall research agenda; see Section 6 with suggestions for further reading.

1.3 Chapter Organization

Our discussion on object properties and management, security issues, and the proposed Global Object Management System (GOMS) architecture is presented in the remainder of the chapter. In Section 2 we discuss the object itself and focus on the properties (2.1), as well as on the object management (2.2). We also discuss and create an advanced object design for objects, and offer supporting examples. Section 3 centers on security concerns and proposed object protections. Second

Life (3.2), and briefly its open source counterpart OpenSim (3.3) and Project Wonderland (3.4), are examined as examples of current object handling, and throughout we examine management and security challenges. This discussion leads to Section 4 in which we put forth our suggested architecture. We conclude the chapter (Section 5) with a look to future research milestones and a summary of our lessons learned to date. Virtual worlds and the 3D Web are still in their infancy and demand more research on object handling, encoding, and especially security. We believe the demand for inter-world exchangeability will supersede the current closed world's approaches resulting in additional compatibility and interoperability questions, as well as potential benefits and challenges with interblending cultures and societies.

2 OBJECTS AND THEIR IMPORTANCE

Shifting the focal point from an avatar-centered perspective towards objects requires a well-founded model to encode objects including their properties, security, and robustness for inter-world transfers. Although the encoding of objects and their interdependencies is the subject of several standards and publications, we outline generic properties and influences on the objects and their management which have to be considered in a secure environment no matter what the encoding (Section 2.1). This model inevitably demonstrates potential security risks without reducing arguments for specific standards or shortfalls of other approaches. In addition, we expand the object-oriented concept where all information and functionality is part of the object itself and not induced by the environment. This facilitates the integration of security provisions and enhances the portability and compatibility. The described model is used in Section 3 to analyze different worlds, and is represented in our discussions of Second Life, OpenSim, and Project Wonderland. However, the model could be applied to any virtual world's objects.

2.1 Object Properties

Information describing an object is referred to as meta-data (Baca, 2008). Dublin Core, EXIF, or XMP are a few specifications to annotate objects like websites or images with further information like size, format, or content; see Wikipedia (2010) for a current list. For 3D objects, standards like X3D (Web 3D Consortium, 2010), O3D (Google Code Labs, 2010), XML3D (Slusallek, 2010), or COLLADA (COLLADA, 2010) provide elements (e.g., XML-tags) to describe properties in the object itself. Here, we discuss the objects from a formal perspective with respect to required functionality incorporating elements from various sources (c.f., in virtual worlds, on the Web, or in applications), but neither from the perspective of virtual worlds nor including a broad coverage of elements. The proposed list is not complete but does demonstrate major categories that need to be adapted within an object and its anticipated application in order to work within our proposed architecture.

We propose that all objects can be classified by their properties into the following categories: *metadata*, *physical*, *appearance, function, environment*, *relation,* and *social*. Figure 1 represents these categories in a general sense; Table 1 describes the categories in more details.

Depending on the environment and usage of objects, further properties need to be added; requirements of being within a certain proximity of the object, having a certain state to interact with an object (e.g., avatar has to sit on the driver seat to operate the vehicle), having certain qualifications, and so on. ***Interaction*** of objects is encoded using the *relation*, *function*, and *environment* properties and allows for dynamic systems; i.e. by providing a server based semantic network for the object relation. Interaction as well as access are encoded in the object and therefore

Figure 1. Properties describing the object. For each property, only examples of elements are shown

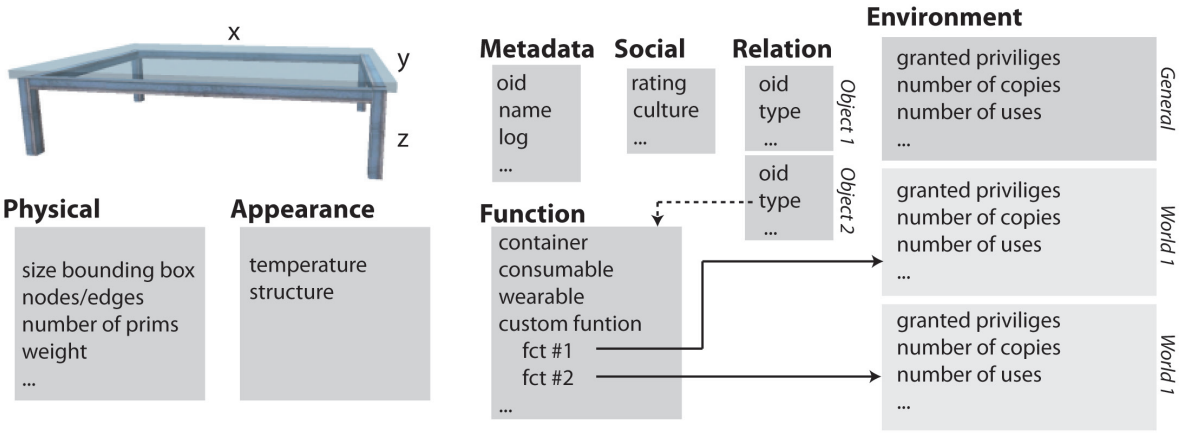


Table 1. Short explanation of the object properties

Property	Description
Metadata	Further information describes the object in more details, which cannot be assigned to other categories. This is general (e.g., name, date of creation), context-dependent (e.g., e-learning, specialized application), or classification (e.g., keywords based on ontologies, field of study). With respect of improving interaction of objects, linking external information resources (e.g., databases, knowledge information systems) to define proper reaction on the behalf of the involved objects are intended; see also the description for *interaction* following the table.
Physical	Objects used in environments that represent (virtual) physical spaces, such that physical properties reassure adequacy of the objects in that environment; e.g., burning torches require oxygen and even virtual life might need food to stay alive. Proper physical behavior requires parameters, e.g., to detect collision of objects (*size*), to determine the capability of floating (*density*), or balancing objects (*weight*). Technical parameters like number of prims, nodes, or surfaces allow an estimation of the complexity and, therewith, the applicability.
Appearance	The object's representation for the avatar; e.g. material with its sensory perception and other haptic experiences. Appearance and physical properties are discriminated as appearance is related to the user (represented by the avatar) and physical to the environment; whereas avatars are considered as objects under the direct users' control. Precisely, properties encode, e.g., observed temperature (based on environment or self-regulated like an oven), structure (rough surface vs. smooth), interpretation of objects being worn (costumes).
Function	Specify how the object can be used by other objects or avatars; e.g. container to store other objects, vehicle for avatars, or food to eat for more energy. Besides some predefined attributes like wearable or consumable, custom functions are the key to an unlimited (interactive) virtual world. Interfaces (*sensors*) monitor the environment for occurring events and trigger assigned functions; e.g. collision, touch by an avatar, or progress of time.
Environment	Link object properties to an environment and need to be specified for each object instance throughout the full lifetime. Examples are granted privileges to access, modify, or transfer the object on individual or group level, number of allowed copies, or restrictions where and how often the object can be used (required, e.g., in virtual games where player buy food items to regain energy for their avatar by using the object in parts or completely fall as well). In addition, the access rights for functions (see *functional properties*) follow the same idea: the object grants (or denies) access to roles, groups, or individual users depending on environment as well as other attributes like time, cost or ownership of objects; see also *relation.*
Relation	Define relations to other objects and used mainly to encode possible interaction. Relations are encoded comparable to semantic networks, where the type of relation as well as the related object is given. For example, a door requires a particular key; it would be specified here with the type *open/close* and the object identifier of the key. Instead of relations to specific object, we could use concepts: A wine bottle requires an object representing the concept of a *drinking container (glass)* for the relation *pour*.
Social	Describes the context of objects; e.g. rating and cultural adequacy. Social properties are required to prevent conflicts where objects should not be presented to certain cultures or religions, or being next to another object; e.g. the kindergarten next to the red light district would be a bad choice. The terms describing context should be mapped against ontologies.

provide the foundation for an inter-environment tool.

Unfortunately, current environments use proprietary encodings and, therefore, are not compatible. For example, there is no common language among all worlds; Second Life uses its own script language, Linden Scripting Language (LSL) with scripts being stored as part of the objects (Linden Lab, 2010a), while Project Wonderland does not yet support in-world functions, but uses precompiled cells on the server-side (Retha, 2008). As a result, functions are linked to their (compatible) environment. Later alternatives could include a code conversion or general scripting language with distinct APIs for each world (i.e., internal object support). Another weakness with respect to security surfaces in Second Life. Access to functions is controlled by the function itself with implementations for each role. Instead of bundling multiple roles into one function (e.g. touching an object causes different reactions for creator and general users), the execution right should be controlled by environmental properties and environment specific grants for each entity.

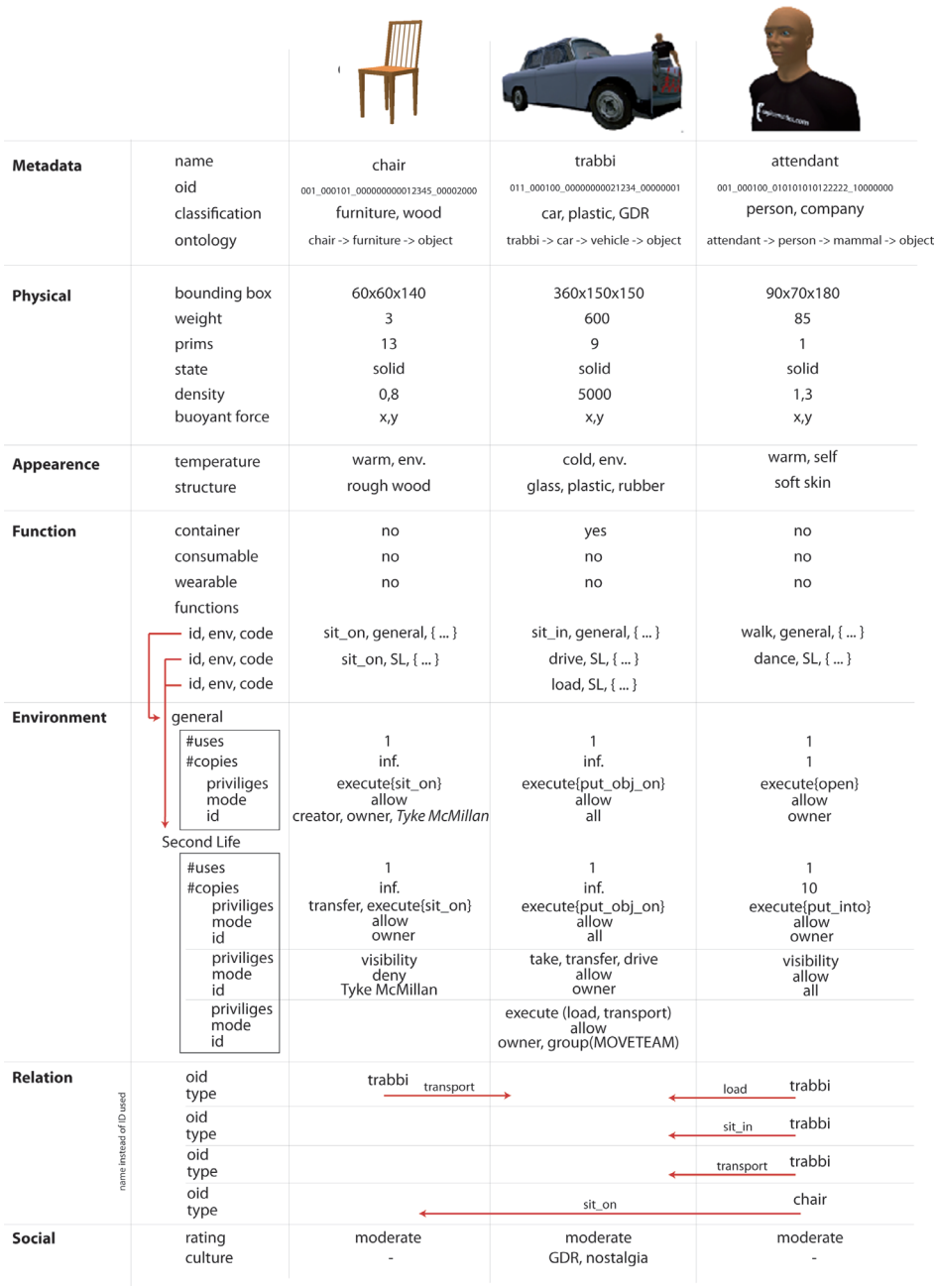
In addition to compatibility problems among the environments, we need to address the relational complexity between objects; i.e. as the information is stored within the object itself. To limit the object's size, only non-general relations or relations concerning the main purpose of the object should be stored in the object. As a general rule for every function involving two or more objects, there is also a relation defined. That is, for example, the key to a door or specific tires for a vehicle. For the sake of generality, external semantic networks with allowed interactions between objects are accessible by the object. Note that the security of a system is increased if object interaction is restricted to semantically correct ones, assuming that objects are specified correctly (c.f., Section 3). Another important application of relations is about referring to alternative (e.g., other media formats, alternative representation) or modified objects (versioning) (c.f., Section 2.2).

The proposed categories with exemplary properties are visualized for three different objects in Figure 2. Although demonstrating the idea of how to encode objects, we do not intend to be accurate in the syntax; i.e. neither the parameters for functions, fully specified relations, nor the reference of relation types to functions are shown for the sake of readability. Properties in the first two categories should be self-explanatory. For example, the property temperature indicates if objects are heat-sources or depend on the temperature of the environment. The car (*trabbi*) is ambivalent and could be either dependent on the environment (parked car on the street) or self-heating (engine and a/c is running). Here, we declared the *trabbi* to be the first case and would model its components (engine, a/c) to be heat sources, with both objects being related to the car in the category relation. In environment, the overall number of *trabbis* is unlimited (inf.), although some virtual worlds like Second Life explicitly restricts the count to 50 to increase its value. Moreover, defined functions, as well as general operations are granted or denied to entities: the *chair* is not visible to the avatar Tyke McMillan. In Second Life the car can only be driven by owners, whereas everyone can sit in it. Finally, relations are defined; i.e., the object *chair* can be transported by the object *trabbi*, and the attendant is able to load the object. Note that the object *attendant* does not have a relation to drive the object *trabbi*, therefore we assume either that the *attendant* cannot drive the *trabbi* (even in case of being the owner) or that specific rules in the server-based semantic network exist on a general level to restrict this relation.

2.2 Object Management

Over their lifetime, objects can face several risks from multiple sources that target flaws in the object design, application, and handling. The best analogy is found in the software industry, where developers try to stay ahead of hackers by regular software updates to fix security breaches

Figure 2. The overview shows three different objects with their properties as well as the interdependencies encoded in the property relation.



(resulting from development errors) and adapting to changed environments (new technology). To improve security, we investigate relevant factors regarding the object management and increase awareness of possible issues about security while demonstrating the importance to have an object-oriented design in a multi-environmental context to include identity, transfer between two or more

entities, visibility, or access. We are aware that the list is not complete and specialized worlds or applications require additional parameters, which have to be carefully considered in the same context with respect to security.

Object security originates in the system design and every non-considered possible breach or missing information can later be used to manipulate the objects, their relation, or interaction. The robustness to new contexts is important as a change of context might reveal information or grant access to invalid entities in a multi-world setting. In addition, the design determines the versatility (to include the combination or communication across environments) and API of objects. Of course, another risk is the user itself. The object needs protection against manipulation, distribution (creating copies, transfer), or loss. The entertainment industry constantly attempts, with doubtful success, to limit the number of digital copies for music and movies using special codices, DRM (Digital Right Management [Zeng et al., 2006]), or specialized software. Although not a focus on this discussion, DRM is a good example for existing secure technological solutions and their limitations. Additionally, trusted computing achieves a sophisticated level of security by involving the whole system; the application is distrusted by the general users as their decision is passed to others (Kabus, 2009). It also increases administrative overhead in case of multiple devices or change of hardware and software. We must learn from past mistakes and we believe our framework embodies these learned lessons in its approach.

In the following sections, we identify several factors influencing object design and its security. Protecting objects requires knowledge of what to protect, where objects are applied and what kind of interaction is allotted. We discuss factors like identity, privileges, transfer, or tracking from an object-oriented perspective. Even though the list encompasses the most important items, we are aware that it could be extended in order to consider other models to encode objects.

2.2.1 Object Identification and Privileges

Objects require unique ***identification*** on a global level. In Figure 2, we used Unified Resource Identifier (URI, Berners-Lee et al., 2005) with the scheme http://obj_id.net/objid to reference the object (domain, global unique identifier). Additional possible encodings are the country code of the creator (3 hexadecimal digits), unique identification of the original environment (6), time stamp (15) and unique object number in this environment (9): *objid*=001_000100_01 23456789ABCDE_01234567; see also UUID (Universal Unique Identifier, Leach et al., 2005) or DOI (Digital Object Identifier, International DOI Foundation, 2010; commonly known in the research community for bibliographic references) for schemes to code persistent identification.

Availability describes the quantity of objects (overall or within an environment), while ***uniqueness*** guarantees that two different objects never have the same identifier. While availability is merely verifying that the number of existing objects is not above a given (global) boundary, uniqueness raises certain (philosophical) questions: what makes an object unique? What qualifies the assignment of new identifiers? Are modified objects new unique objects with new identifier or can we consider them to be the same as before? Nevertheless, we suggest a simplified approach: An object is the same as long its physical structure is unmodified, e.g., parts, edges, nodes, or surfaces are constant in number and position. On the other hand, any change of the physical structure results in a new object, whereas the original is copied for references and versioning (c.f., Section 2.2.2). For example, if a car is painted green, it is still the same object. If the car is modified by a new body, we consider it to be a different (and, therefore, new) object. Both objects store references to each other for traceability, such that modifications can be resolved to the very first original object.

Access and interaction depends on the ***privileges*** granted to entities. For conformance, interchangeability, and security, *basic* privileges need to be globally defined, whereas the environments guarantee the compliance of privileges and their correct implementation (c.f., Section 3.1). We distinguish *basic* (e.g., transfer, modifying, touch, and visibility) and *user-defined* (e.g., driving or using an object in a certain way) privileges in our architecture because the latter usually require a specific function to define the privilege implications in the environment. Object privileges are specified by parameters such as:

1. **Identities**: Comma separated list of identities that defines the entities for which the given privileges are relevant. In addition to entities, roles can be used (e.g., *all*, *none, creator, owner,* or *user*).
2. **Privileges**: Comma separated list of privileges based on a global dictionary.
3. **Allocation**: The keywords *deny* or *allow* either grant or revoke the privileges.
4. **Time Validity**: Time period during which the privileges are granted or revoked. For example, major content is only visible for the group *adult* between 10pm and 4am.

Each custom function (property *function*) as well as environment (property *environment*) has its own set of independent privileges. With respect to administrative overhead, objects assign privileges to roles or groups rather than individual avatars or objects. However, in cases with high demands on personalization or security, individually granted privileges are the first choice. If privileges depend highly on the environment and offered object functionality, a basic set might be limited to privileges like access, transfer, visibility (e.g., research department, where visitors are not allowed to see everything), mobility (allowed to move the object), delete, take, or modify the object.

Objects allow different ***roles*** (e.g. *creator*, *owner*, or *user)*. In contrast to the general allocation of roles to users and groups, objects are treated as equivalent and are able to be creators, owners, or users (e.g., an aerial scanner which creates and sends out drones to collect data). In this case, either the creator of the scanner (scanner follows instructions implemented by its creator) or the scanner itself (fully in charge of the process [e.g. autonomous and intelligent software agents]) is the creator of the drones.

The argumentation is diverse if we look at multiple entities being creators for the same object. How would we share responsibility? Are creators responsible for their part or the whole object? Can every creator modify all parts or only their own? Can one creator restrict the access for other creators of the same object? It might be possible to model multiple creators but for practicality and acceptance, we need a simple and intuitive concept. For that reason, we allow for the role *creator* only one avatar or object. To allow collaborative work, the responsible creator needs to assign privileges like *object modification*.

Every object is owned by (multiple) entities. Note that multiple owners share the object equally with the same privileges. For example, revenues or votes for decisions are equally distributed to all owners rather than using different shares similar to shareholders in publicly tradable companies. If such functionality is required, it can and should be implemented as a custom function. Comparable with reality, multiple ownerships are valid even for objects taken into the inventory of one avatar; the access on the other hand might be restricted. The first owner is always equal to the creator, who also sets the initial privileges. Whether an owner is able to change or pass privileges depends on the granted rights.

Last, we have the role *user* to control interaction with the object based on privileges and functionality. This role is used to explicitly limit the access to specific entities; not specified ones are not allowed to get privileges granted. Only owners are special as they cannot be excluded from the role user.

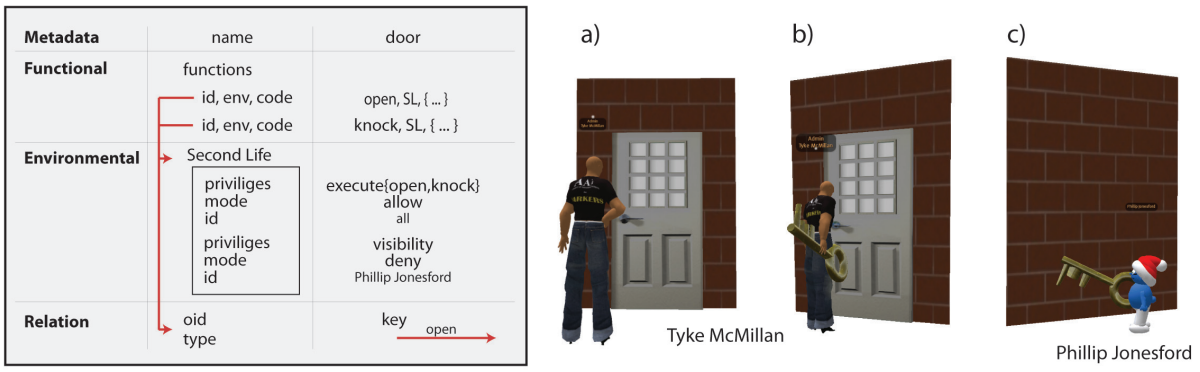
2.2.2 Object Access and Transfer

Privileges grant ***access*** to objects and define the interaction, which are either default functions or functions defined in the object (*functional*). Besides using privileges, the access to objects can rely on other objects being owned by the entity; e.g., key cards to open doors. The relation of objects can be defined in two ways: by object identifier or object metadata. The first approach is about higher security as the creator defines the *lock* and *key* by object identifier (key) in the relation properties of the lock; see example in Figure 3. Access to the lock is only possible if the key is owned and carried (on the avatar or in the inventory) by the authorized entity. (Unmodified) copies are also eligible as they are cross-referenced with the original; here the creator keeps responsibility for allowing copies. Using common metadata description is less secure as other avatars can create objects with matching properties. Examples for scenarios are clothing, extensions for rooms, or food items in role games.

Especially in scenarios where the objects are subjected to high security, invisibility to non-authorized users is the safest solution (Wright & Madey, 2008a). The idea of granting ***visibility*** rather than access relies in the general server-client-architecture, where non-visible objects are not transferred to the client and, therefore, are safe against most attacks. Another example how visibility can be used, is set in a theater. While objects on stage are visible but not accessible, stage-hands need to (re-)build the stage setting without being visible. This can be solved using different spaces; that is, the visibility of the avatars (being objects) and all objects for the next scene is denied for everyone but stage workers, who setup the stage. During a break, the visibility is flipped with the current stage.

We distinguish two kinds of ***transfer***. ***Passing*** is about changing the ownership by giving objects to one or more entities. ***Copying*** is about replicating an object and passing the new object. A change of ownership denotes several considerations about rights, privileges, and properties. It goes without saying that with lacking privileges or access to the objects, neither passing nor copying is possible. First, the entity triggering the transfer must have adequate privileges and be the owner. In case of multiple owners, it has to be sufficient that one owner passes the object without getting approval of all. The trust for correct decisions was approved by granting the privilege in the

Figure 3. Demonstrating object visibility and access. In Scenario a), Tyke is in front of the door but does not have the key. Therefore, the door is looked. In Scenario b), Tyke has the key and can open the door. In Scenario c), Phillip has the key, but the visibility of the door is not granted to him and, therefore, the key not usable to get on the other side of the wall.

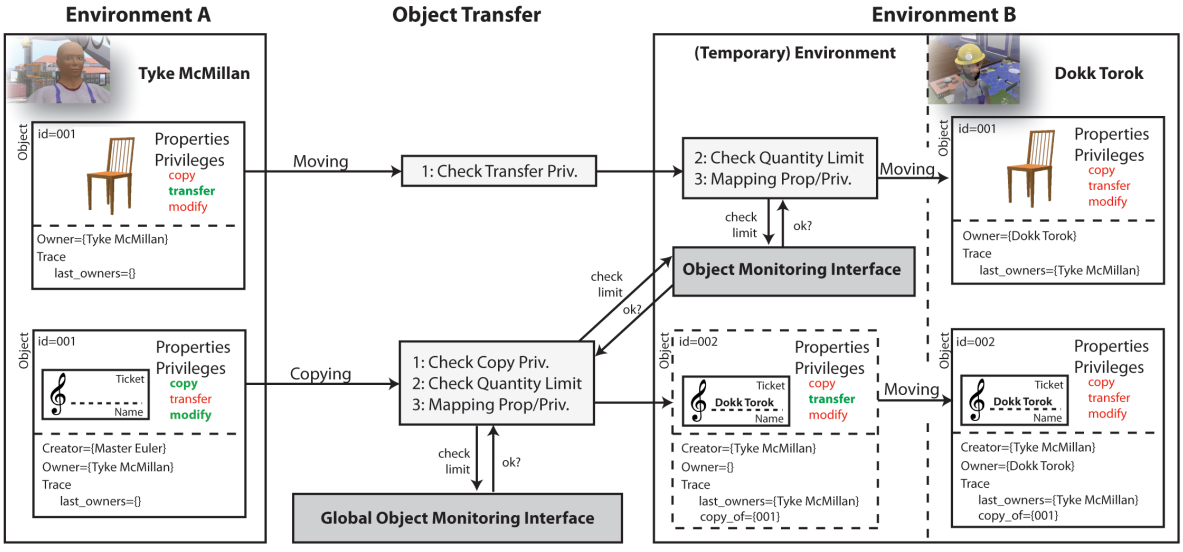


first place. Note that it is technically possible to ask all owners before transferring the object, but would result in long delays even if everyone would agree on the transfer; e.g. delays through operative tasks, holidays, or sickness. Second, the mapping of properties and privileges for the next (new) owner needs to be specified. That is, the current owner has to define for every property or privileges how these are passed and if the new owner is able to change or pass these to the next owner. For example, a shop owner is allowed to personalize and sell tickets (even though he is not the creator) to someone else, who is not allowed to modify the ticket afterwards, e.g., date or category; see Figure 4.

Copying involves the creation of new objects and, therefore, requires an additional consideration: who becomes the creator of the copy? Basically, it is either the creator of the original object or the entity who initiated the copy process. In our opinion, the second concept is superior as the latest responsibility for the object is visible; see also object traceability to guarantee references to previous creators. Note that the access for the original creator has to be granted through privileges as well. This allows, given the right privileges, to prevent even updates or other access (like revoking the object) by the creator and, therefore, keeping object static; e.g. in test scenarios where updates would change the outcome. Another plausible idea might be an empty creator field to differentiate copies from originals; if the creator is kept in the trace. Copying also requires verification of availability.

1. **Unlimited Object Quantity, Restricted Environment:** The entity requests to copy a limited object, which is either authorized

Figure 4. Transfer of objects. In the upper part, Tyke McMillan transfers an object to Dokk Torok who is not allowed to pass it further on to others. Both have no privileges to make copies or perform modifications. Quantity is verified locally within the new environment. The lower part demonstrates copying, where Tyke McMillan owns a ticket that can be copied but not given away (that is, the blank ticket stays with Tyke McMillan). Tyke has the right for modification, i.e. inserting a name on the ticket. Quantity is checked on global and local level as a new object is created. Note that functions (inserting names while copying) and properties (creator, trace) are not shown with all details.



or denied based on the total count of currently existing objects in the environment, including inventories. Therefore, environments have to monitor object counts and apply control mechanisms like *first come, first serve* or *hierarchical structures,* where certain entities overrule concurrent copy requests; e.g., requests by avatars outvote requests by objects.

2. **Limited Object Quantities:** Albeit the object knows its own maximum number of existing copies, a superior authority needs to monitor the overall existence of limited objects in all environments. In general, a trusted server is used, where copy requests are handled similar as in environments.

In both cases, we need to discuss the influence of the previous owner or (original) creator. In several scenarios, influences are welcomed by new owners (e.g., updating functionality) or previous owners (e.g., retrieving rented objects after a predefined time period). Many recent cases illustrate that buying objects in Internet stores do not imply unrestricted ownership, such as Amazon deleting E-Books on the Kindle (June, 2009), or Apple reserving its right to access and revoke applications on the iPhone (von Lohmann, 2010). The (negative) feedback towards (unknown) surveillance by companies/governments enforces exigencies of rules for inter-entity access to be followed: transparency, revocation of access privileges, and traceability of all performed procedures.

The object transfer is also restricted by rules (e.g., no allowed copies), technical properties (e.g., the object format is not compatible with another environment), or the environment itself (no export from closed environments). In addition, the properties can change over time and/or environments; examples are price or age of the object influencing the functionality. For the sake of interest, we should briefly discuss the relevance to use standards for encoding object details. Without questioning, standards are superior to proprietary formats in case that objects are not supposed to be limited to one platform; the number of import and export filters is reduced, redundancy and transfer errors are avoided, structural errors of the format are found faster, and documentation and tutorials are generally available. Over the last years, several standards, specifications, or proposals for standards were developed, for the purpose of object storage or security, the format is not relevant and we consider the objects to be binary files.

2.2.3 Object Information, Tracking, and Quality

The ***Metadata*** represents additional information. Depending on the virtual environment it can be generated automatically or has to be specified by the entities themselves. Further information about the object is given by building ***inheritance trees*** where properties (e.g., functional, environmental, and relation) are selected and merged for child objects. New objects can also be created by combining objects. In contrast to inheritance, the objects are kept as they are and combined to larger ones; i.e. by defining their relative position to each other as well as the interaction of components. Examples are the combination of vessel and crane (ship being able to (un)load storage without external equipment) or tires for vehicles, where the relative position is as important as the type and the granted rights for tire changes.

Objects need and should (if not otherwise specified) ***trace*** all occurring changes, modifications, and relations to other objects. Generally speaking, the information about the object is stored as part of the object; whereas the environment is responsible for maintaining the trace. Note that the mechanism has to be used with caution, as the trace might flood the object and cause huge object size. Versioning corresponds to keeping track of copied and modified objects. The trace within objects can be used to recreate every previous version; assuming that the trace is copied for

every new version of an object. We discriminate the following information types, whereas further are defined by each environment. Information is stored with timestamp, environment, and involved entities.

1. **Trace:** Keep trace of originating object in case of copying, change of owner, or modification of major properties (e.g., regarding grants or functions). The information here is, e.g., required to trace object history, versioning, inheritance, or ownership.
2. **Function:** Allow functions to write to the trace log.
3. **Detail**: Keep track of any modification of the object; e.g., movements, interactions with other entities, or modifications of less relevant properties.
4. **Debug:** Include further information to debug specific scenarios or tracing errors. Here, information about the environment is included to reconstruct specific settings.

Persistence is the "characteristic of data that outlives the execution of the program that created it" (Wikipedia, 2010). To transfer this to our scenario, the secure and unchanged storage of objects has to be guaranteed. First, each environment has to take care of this, as well as the Global Object Database; see Section 4. This requirement is in line with specifications for databases. On the other hand, virtual objects are part of shared environments with interaction while the owner is offline. Assuming correct grants, the object changes are in consent with the owner. Changes are communicated and stored in form of single messages (based on time intervals or occurring events) or cumulated reports being send to the owner. If an object state needs to be kept, either the grants must be very restrictive or replicated to a secure environment.

Finally, the ***context*** and ***quality*** of objects is specified by several properties. The context is characterized by, e.g., community appropriateness, compatibility, and collaboration, whereas the quality has to be seen with respect to the context as well as the user's expectation. Security in this context comprises the *price-quality-relation* such that the risk of overpricing or deviation in promised functionality is reduced; i.e. as uncountable projects (starting with Second Life to Linux to Gimp to OpenOffice) and repositories demonstrated the effect of motivation and believe in sharing by containing high-quality objects being available for free download.

Community Appropriateness accentuates the context of objects in relation to specific communities. Objects placed in an environment by the user require certain guarantees about the neighbors' behavior with respect to quality, kind, or appropriateness. For example, the virtual representation of a company has a showroom with replicas of the real world toys being produced. It is seldom appreciated if, e.g., a strip club or shop for nude skins opens in the neighborhood. Without further discussion where to draw the line between appropriate and inappropriate, certain mechanism should allow guarantees about (visible) neighboring objects. Another property for community security is about internationalization and consideration of cultural aspects of the intended users. First, objects as well as their human readable properties need be available and extensible in multiple languages. Second, the different interpretation of an object from culture to culture needs be considered at all times, either by warnings, limited access, or masking for certain users; e.g. not showing objects in a store by hiding or replacing.

Compatibility has to be approached by three different directions: (1) Can objects be combined and work in a given environment (intra-compatibility; e.g., combining two objects as a group and forming a new object)? (2) Can an object be used in any environment without losing its functionality? (3) From a plain technical perspective, can the encoding of object information and functions be readable and either be applicable or executable

(we believe so). While general properties and structure can be transferred to different formats without a problem, it might not apply to functions where inter-language transfer is rather complex and involves high risks of errors.

Collaboration on the one hand is the communication between objects (e.g. interaction between agents) and describes the way objects can handle given problems and tasks together. On the other hand it stands for the possibilities of one or more avatars editing (e.g. building, designing an object) or using (e.g. driving a virtual car with both avatars being able to control the car) objects simultaneously.

3 SECURITY ASPECTS

3.1 Risks and Protection

The previous section illustrated key requirements for objects in a multi-virtual world setting. Obviously, every item on the list is vulnerable to harm, either via criminal intent or by accident. Even though criminal minds will find ways to overcome almost any security barrier given time, it is important to increase the required maximum effort, while not interfering with the handling, features, preventing the application from effectively running. As mentioned previously, it is not our intention to describe the technology, but to increase the risk awareness in order for users to think about precautions. In the following section, we browse through all object related aspects and describe options for protection. Note that this is by far not complete and does not consider all components to initiate an attack. For example, we do not discuss visualization protection, where the object can be extracted from the graphics card memory, scanned from the monitor, or intercepted by knowing the rendering algorithms in open source software. If the viewing of objects needs to be secure, closed (and proprietary) software is needed on protected systems. Figure 3 shows an example for a shop, where intruders manipulated the items in various ways.

However, objects need to be transferred and stored on local media for individual repositories in order to maintain backups of objects and their associated intellectual property. Therefore, our main concerns are related to (illegal) copies and modifications of digital objects. This is comparable with DRM in the music industry or eBook handling, where companies need to prevent illegal copies by means of linking (to specific hardware) or determent (inscription with traceable information about the buyer). For 3D objects similar ideas can be used to integrate the cases above for a secure system.

3.1.1 Object Identification and Privileges

Observing a specific environment, the ***uniqueness*** of ***identifier*** and control of objects is done efficiently by internal object servers and databases. With multi-environments, we require global servers and either standards used by all environments or dedicated mapping functions with respect to objects and their properties. The global object server is used to register new objects, verify identity, validate the integrity, and protocols in their location. In addition, asymmetric encryption (key pair; c.f. Furht et al. (2005)) is used to link the creator to the object as well as guaranteeing the unsophisticated content and meta-data. Here, the hash code and certificates for the environment (virtual world, application, and inventory) are used on trusted systems with respect to security (c.f. Torres Padrosa (2009) with focus on DRM). Like DRM, the platforms are licensed and verified for modification (e.g., malware) before objects are imported. In addition, digital objects are encrypted to prevent unauthorized copies and to allow distribution on public channels or media. Encrypted objects are matched with certificates based environment and users and can include further limitations like export or time of avail-

ability. Certificates restrict the usage of objects and check the adherence of ***availability*** by limiting the issuance.

3.1.2 Object Identification and Privileges

Observing a specific environment, the ***uniqueness*** of ***identifier*** and control of objects is done efficiently by internal object servers and databases. With multi-environments, we require global servers and either standards used by all environments or dedicated mapping functions with respect to objects and their properties. The global object server is used to register new objects, verify identity, validate the integrity, and protocols in their location. In addition, asymmetric encryption (key pair; c.f. Furht et al. (2005)) is used to link the creator to the object as well as guaranteeing the unsophisticated content and meta-data. Here, the hash code and certificates for the environment (virtual world, application, and inventory) are used on trusted systems with respect to security (c.f. Torres Padrosa (2009) with focus on DRM). Like DRM, the platforms are licensed and verified for modification (e.g., malware) before objects are imported. In addition, digital objects are encrypted to prevent unauthorized copies and to allow distribution on public channels or media. Encrypted objects are matched with certificates based environment and users and can include further limitations like export or time of availability. Certificates restrict the usage of objects and check the adherence of ***availability*** by limiting the issuance.

Granting ***privileges*** follows a hierarchical top-down order. The ***role*** creator has all privileges by default and can appoint new owners and users including their privileges. Based on the privileges, owners are allowed to add additional owners and users, while users are generally not allowed to add additional entities to the object. Note that it should not be technically restricted, but the owner or creator should decide on this in principle. Another exception to this approach is that the owner can reduce the privileges of the creator to forbid access or visibility. The privileges are not defined by certificates but stored within the object. Encrypted hash codes are used to check for modification. Keeping the privileges accessible for inspection has advantages user with sufficient rights can be contacted and misuse is reduced by visibility of names. The security relies on the encryption and the integrity of the environments. Certificates are used from a global server, which could also be used for verification against a central database with all privileges.

3.1.3 Object Access and Transfer

Again, we have to distinguish passing and copying. ***Passing*** implies an authentication of the user and its certificates (i.e. being the owner and allowed to transfer the object) as well as a substitution of ownership and location of the object. If ownership is passed, the existing certificate for the original owner is invalidated such that objects cannot be used anymore; e.g., import into and use in (uncertified) environments would fail. ***Copying*** initiates the creation of a new object with instantiation of properties. Most important is the verification on allowed availability of copies. Here, the global server controls to total count as objects only know about the limit but not the concurrently existing objects.

Legal copies can be secured and monitored. However, what about illegal processes like stealing, plagiarism, alternating? Objects are digital assets and can simply be copied. Therefore, if the environment is not protected or objects are stored on public servers, stealing equals a simple copying process. The encryption (i.e. if the environments are running on trusted systems) of objects and certificates secures the objects sufficiently, such that hacking is not in proportion to the value of the objects and they become a less attractive target, or perhaps become more attractive yet unobtainable. On non-secure systems, object information might

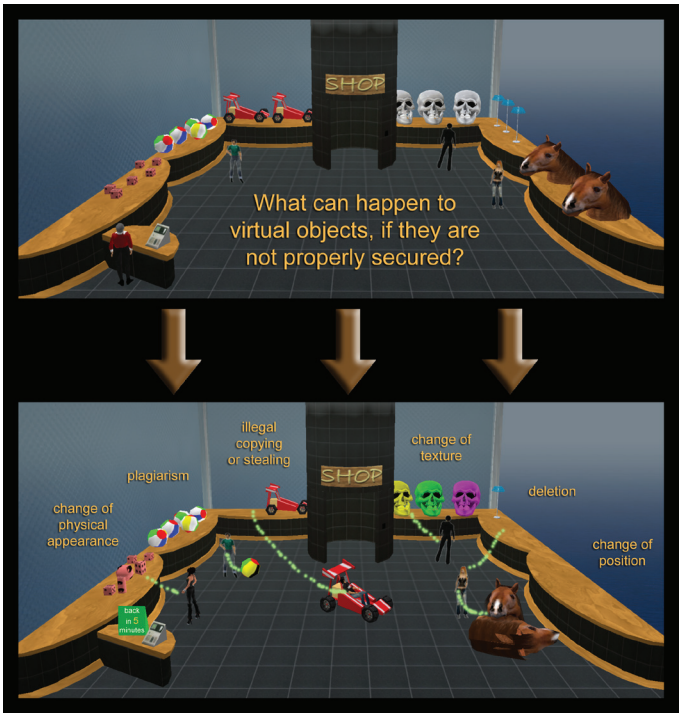
be extracted while being processed so there is not much that can be done about this but to require a secure environment before sending certificates for encryption. The same applies for alternating between types of systems, as it requires decryption and verification using the certificate.

Plagiarism is another issue as someone rebuilds objects or distributes slightly modified objects as their own. Here, watermarks (also called tattoos) can be used that are integrated into the object and allow identification and therefore tracking of the original creator. The disadvantage of watermarks in 3D objects is their robustness compared to watermarks for images (Eshraghi & Samavati, 2007). We also need to differentiate between private users who are interested in modeling their reality by creating replicas from the real world (and do not sell and distribute them afterwards) and the professionals who build their business on brand replicas; e.g. car or sport brands. We do not intend to discuss laws and existing cases as this is different in almost every country. Nevertheless, on important security concern is to protect objects from unauthorized copying (even though we know from movie and software piracy that the race cannot be won) and to identify copied objects.

3.1.4 Object Information, Tracking and Quality

Modification covers almost all cases shown in Figure 5 but copying and stealing. Protection is only possible, if an intruder has no direct access to the object including its meta-data and properties. Any part that is not allowed to be modified without permission has to be signed. However, if objects can be transferred out of a secure environment (e.g. the users' hard disk), the object content is readable and could be used for a new object. Encryption of the object prevents any access without having the right certificate. Certificates do not have to cover the whole object and might be associated with or

Figure 5. Possible manipulations in a virtual world store, if we do not have a sound security concept



exclude components. For example, a virtual house needs modifiable surfaces to decorate the walls, but perhaps the floor layout cannot be modified.

The major crux with all of this is that every component must act jointly. Transferring a secure object to an (assumed safe) environment also initiates the certificate delivery to decode the object. It would be realistic to hypothesize that the user is neither aware of the ambidextrous behavior nor would he notice the operations on the object. Comparable back doors (as well as other issues like compatibility problems of asset and application) existed with DRM for music and caused its decline: using non-digital media after decoding and compression to dissolve watermarks created DRM-free copies. We have already mentioned the counterpart where programs read the graphic card memory. The computer industry already has different solutions to provide secure platforms, e.g. trusted computing (Trusted Computing Group; 2010) or trusted networks (Open Trust, 2010).

Systems are as secure as their weakest component because someone can enter the system and change behavior. Therefore, for maximum security, we need secure (and trusted) systems. We have covered some issues to demonstrate important issues to be considered in an open inter-world setting. Even though arguments about open source and free objects abound and should be considered, it is necessary to provide protection against users not following a codex, but instead cause harm by stealing, changing or deleting. Even within a secure system, it is always given to open objects and share code and design, and strongly supported by the authors. Nevertheless, companies relying on making money within virtual worlds must have a means to protect assets and intellectual property, in order to have the confidence that they can participate in and thrive in virtual worlds, as well as expand their virtual world offerings.

3.2 Examples in Real Environments

In this section we examine Second Life (and to a short extent OpenSimulator and Project Wonderland) to provide contemporary examples of object handling. We are not going to cover every aspect about object handling but want to provide a rough understanding about what happens in 3D worlds. Sure, there are almost uncountable other worlds (see KZero (2009) and Virtual World Review (2010) for an overview) each with unique object handling and protection. We decided on Second Life as it is the most popular one with respect to in-world development of objects and on the other two as they provide a very good platform to implement our concept. Also these worlds provide opposites like closed vs. open world, entertainment vs. business context, single grid vs. multi-grid, and so on; but this would not be part of the following discussion.

3.2.1 Second Life

Second Life was started in 1999 by Philip Rosedale with Linden Lab and started much of the interest about second (virtual) realities not only in popular media but also scientific journals; cf. (Linden Lab 2010d). The success was primarily related to the immense possibilities for customization of one's avatar, environment, as well as the intricate economic system and ability to communication and interaction with others in a free to form environment unlike games with their codified rules. It should be noted that Second Life participates in many research projects supporting virtual worlds and has contributed greatly to various disciplines in this respect; however, it is still considered a closed world as avatars and content is still not freely exchangeable. Still, we can benefit from a brief discussion in terms of how this virtual world manages its multitude of objects.

3.2.1.1 Object Identification and Privileges

Second Life uses the Universal Unique Identifier (UUID, Linden Lab, 2010b): a 128-bit ***identifier*** following standards by the Open Software Foundation. This approach provides for a large range for UUIDs, as well as rules for creating unique UUIDs, such that the likelihood of unintentionally assigning the same one twice is very rare. Thus, no central server for coordination is needed, providing a less complex mechanism. Using this approach, the closed Second Life system can validate new UUIDs and prevent identifier collisions.

Second Life further protects objects in its system by operating all of the virtual world servers and allows limited access only after authentication. We should note that this approach might be modified and relaxed somewhat with Second Life offering new servers users can buy and use behind organizational firewalls with proposed full access to hosted islands at the administrative level (Linden Lab, 2010c) and continuing development on the Open Grid Protocol (Linden Lab, 2009a). However, this would most likely be cost prohibitive to most individual users.

In Second Life, objects have creators (the very first owner without additional rights compared to owners) and owners; access control for *user* can only be done within scripts. That is, ***privileges*** to functions are part of the scripted function and changes require programming knowledge. Another example is using the internal function to move objects (non-physical ones, as others can be moved by just applying force to them). This feature is either implemented as a function including permission for entities or granted by selecting an option to allow moving objects by anybody.

Another privilege is given through parcel management, which restricts access but not visibility if it is on an island with public access rights. In Second Life, to provide protection (access and visibility), objects require an isolated island not attached to others. The roles and capabilities in Second Life allow specifying users within groups, but with respect to objects, it provides only transfer to groups (and not roles within the group), access to manipulation as moving, copying, editing, and setting the object for sale. Other object related capabilities are indirectly controlled through parcel management, where settings specify who can create new or return existing objects. All of this leads to some particularly challenging security and access issues in Second Life. As described in the following section, the limitations are quiet high and limit the flexibility as well as the security. That is, secure and public areas have to be clearly separated (i.e. due to visibility issues) and user specific access is limited to roles and capabilities or requires specific scripts (compared to user role or having object-related privileges that can be granted to users).

3.2.1.2 Object Access and Transfer

One of Second Life's main features is the economic system, where users buy and sell objects and land. As a result, the transfer of objects is closely associated with sale prices; however, the price does not affect the transfer itself. The transfer is limited to the role *owner* as users are handled within scripted functions and the creator is considered to be owner as well. Privileges for the next owner cannot be specified beyond very restricted limits (Linden Lab, 2009b). The next owner gets either full access and modification rights to the object, is only allowed to create further copies, or has to keep it as it is. The next owner is selected by passing the object to another entity (in Second Life restricted to avatars), whom receives the object with the chosen rights. Second Life does permit an option, where the object can be copied by everybody who also gains full access by this. From a security perspective, this is unacceptable as it provides unlimited copies in an uncontrolled manner.

Second Life tracks three privileges that can be passed to the next owner: 1) modify (change everything about the object), 2) copy (create unlimited copies of an object), and 3) resell (next owner can sell the object). If copy and sell are

activated, the next owner can copy and sell the object as well, causing an uncontrolled distribution with possible enormous economic damage. When a person passes objects without copy permission this implies that the object is given away without any possibility to retake it. Users in Second Life can create temporary objects with a built-in destruction timer and Second Life provides visible options to encode this in functions. Temporary objects and areas that clear themselves after a certain time endangers persistence as the world is changeable without real control; even though the objects are returned to the owners' inventory. Note that scripts have their own permissions similar to object permissions so this framework can be extended here as well.

3.2.1.3 Object Information, Tracking and Quality

Second Life uses properties for objects in order to support the advanced physical engine and provide a true immerse feeling for users. Nevertheless, restrictions apply as some properties are not permitted to be changed to provide more straightforward realistic modeling: Object weight is determined by size rather than being able to specify concrete mass, flexible objects cannot be physical, and objects cannot be connected to others; e.g. connecting a rope to a hook or having a realistic towing bar. Programming manipulation allows implementing these elements but only with detailed effort and advanced knowledge about the system. In addition, objects lack information for retrieval, usage, and relation. Besides object name and description, no further object information can be assigned by developers such as meta-data, cultural, or social appropriateness, as well as (semantic) relationship to other objects. Although verification for object placement in a certain area can be based on age, ethics, or culture on a higher level, by having underlying ontologies or rule-based systems, this is unfeasible because with over a billion hardly-specified objects, object retrieval is comparable to finding the infamous needle in a hay stack.

Please note that users can create objects in-world or with external applications like Blender to allow for detailed object parameters. However, once imported, these objects are stored in-world on Linden Lab servers and come under the same restrictions as objects created in-world; moreover, an export out of Second Life for further modification is currently not anticipated. Applications do exist (e.g., Second Inventory) that permit backups of completely owned objects (with full access rights) in the avatar's inventory to external storage devices, but not of islands or specified objects in the world (even user-owned). Nevertheless, the security is minimal for controlling access to objects and producing (unauthorized) copies because other software can download whole islands even though it violates Second Life's ToS. Another program, Open Source viewer, can access all in-world information to visualize objects and, therefore, can create digital replicas. Currently, there is no mechanism to encrypt objects or require certificates to validate object ownership. This makes the protection of intellectual property a challenge.

Except for the ToS prohibition in Second Life against external copy tools there are no features or controls in place to help creators properly protect their objects. However, Linden Lab is aware of the problem and is considering several changes, especially because everything digital faces the risk of being somehow copied (Linden Lab, 2009c). Some form of an approved rights management program could make large strides towards a safer environment for Second Life creators.

Ultimately, Second Life is a pioneer in Virtual Worlds, but its closed-world system leads to many object management shortcomings in terms of access, interoperability, and security. Objects are for in-world usage with limited inter-world transfer possibilities. Although server-based storage and execution of scripts lends itself to high-level protection, it also leads to a very restricted access to specific object properties. Moreover, security is

not part of the object design (besides the ownership and passed rights for the next owner) but requires individual implementation using internal scripting language. Relation and context-based properties are not implemented and object naming and tagging is not enforced which results in the difficult retrieval of objects. Security can be implemented but requires programming knowledge and loyalty to the Second Life environment as the restrictions cannot be ported to other virtual worlds.

3.2.2 OpenSimulator

OpenSimulator (also called OpenSim) started in 2007 and is an Open Source 3D application server to create a virtual world being compliant with Second Life viewers (using the same protocols for the client-server communication and grid management) and also resembles many other aspects (OpenSimulator, 2010a). Nevertheless, OpenSimulator is by far more than just a re-written Second Life server as it provides a larger functionality, while giving full control over the server through local installations (behind the firewalls of organizations) and unrestricted interfaces to integrate further applications. In addition, the functionality can be extended by loadable modules, multiple protocols providing communication beyond the OpenSimulator grid, users have full control over their objects, and in-world scripting is not limited to one proprietary language but can be, e.g., LSL, Jscript, C#, Yield Prolog, or VB.Net (OpenSimulator, 2010c).

In contrast to Second Life, OpenSimulator can be installed on individual computers, providing the user with all rights for administrating the system; including access to the virtual objects for, e.g., backup. The same advantages are given in the so-called grid-mode (depending on the access rights assigned to the individual users) where multiple instances are connected to a larger virtual world and, from a technological perspective, several servers are used to distribute the required workload. Individual servers are set up for the management of users, grid, assets, inventory, messaging, and regions. It is anticipated (and done in first experiments) to connect the OpenSimulator grid to the Second Life grid and allow traversal of avatars from one world to the other (Linden Lab, 2008). But so far, the teleport works just for the avatar without any objects including the appearance of the avatar itself. However, object transfer between OpenSimulator and Second Life is subject of further research; a detailed description can be found in Sequeira (2009); whereas security is not the main concern in this work.

The similarity (and compatibility) of OpenSimulator to Second Life with less administrative limits and less costs for maintaining land, OpenSimulator is an interesting test and development platform for users as content (objects) can be exported to Second Life; i.e. if LSL was used as the script language (Winkler, 2010, Vilela et a., 2010, Novak, 2010). Note that OpenSimulator is still development in process and while it proofed its robustness in real-life projects, many aspects like full support of LSL are not yet implemented to their full extent. From a research and educational perspective, the missing economical support might be of little relevance, but companies with ambitions to make revenue are not able to use OpenSimulator for their productive installation. Another problem in OpenSimulator, which might be the result of the current status as alpha version and still in development, is the missing terms of services and the therewith the problem of defining and claiming copyrights.

The security of objects is similar to Second Life. Objects are transferred to the client for visualization (while objects with no access are kept on the server in Project Wonderland) and allow access to the object structure through this. Encryption and further protection through tracking and watermarking are mentioned as possible extensions but not yet specified with any means. Note, that the storage of objects is different in Second Life (file system) and Open Simulator (database). The architectural design of OpenSimulator provides a

great basis for concepts like we suggest in Section 4 as the server structure already separates different responsibilities.

3.2.3 Project Wonderland

Project Wonderland is an Open Source collaborative 3D virtual world implemented in Java and designed as a server-client-architecture. Until 2010, the project was funded by Sun Microsystems (Oracle, 2008) who intended to develop a robust, secure, reliable, scalable, and functional environment for organizations to improve communication among distant partners, but also to work collaboratively together by having all common desktop tools like OpenOffice available. The funding was stopped during the acquisition of Sun by Oracle and is continued (by the same team) as Open Wonderland (Open Wonderland, 2010; Korolov, 2010) with the same focus.

Project Wonderland is built on top of the 3D engine jMonkeyEngine and the game server Project Darkstar. The virtual world is internally described by a (3D) scene graph with the node *world* as the root and all other *cells* hierarchically organized from the root down. *Cells* are the objects described as the volume they take in the virtual world. A cell can contain further cells and can be positioned, rotated, or scaled. To each cell (which could be a TV set), capabilities are assigned to describe what can be done with the object. This is comparable to the functions used in our object model. The cells' initial state is stored in a hierarchy of XML-files (also called Wonderland File System), which describes the cell structure and the containing components, modules, and artworks. The cells can be created by every user and dynamically loaded into the world. Simple editing is possible since version 0.5, such that location is not necessarily set in the XML file but using graphical interfaces.

Security is a major aspect for Project Wonderland. The Open Source approach allows an installation behind corporate firewalls and the extensive APIs can be used for integration of the existing authentication and identity management of the business. The objects are secured by (hierarchical) access control lists to control visibility, interaction, and editing. For Project Wonderland, a DAC implementation with focus on simplicity exists. The *Discretionary Access Control* (DAC) restricts the access to objects for users or groups, similar to the UNIX model (Wright & Madey, 2008); i.e. passing the rights down in the hierarchy. The access permissions can be passed on to other users. WonderDAC has three user groups (*owner, group, everyone*) with two permissions (*interact, alter*). The authors keep the complexity (and therewith the granularity for access rights) low in favor of acceptance. Thus, objects with no interact permission will not be visible to avatars. Note, that without permissions, objects are not submitted to the clients to increase security. Further security features like encryption or prevention of stealing objects is not (yet) implemented, even though DAC can be used to restrict access by unauthorized users.

Second Life and Project Wonderland are targeting different users. While Second Life is focused on in-world activities using the provided tools, connectivity to external applications is very limited due to the restricted interfaces. Project Wonderland, on the other hand, allows import of most formats and (almost) unlimited extensibility by further modules. The main target groups are (business) teams working on real-life projects like meetings for brainstorming, presentations, or product development. To provide a virtual counterpart for an office setting, Project Wonderland allows inclusion of desktop applications like OpenOffice or Internet Browser. Second Life also allows the projection of web sites (including interaction), but is not supporting collaborative work as the interaction is local to the client.

Creation or modification of objects in-world is not possible and reduces its value especially for users without experience in using external programs for modeling. Second Life and Open-

Simulator already have full editors allowing everyone do be creative. Same applies to in-world scripting, which is currently in early development in Project Wonderland but will provide a more flexible way of extending the functionality compared to Second Life as different programming languages without restrictions will be supported. Other missing features are, for example, personal inventories, exchange of objects between avatars, or an economical system and in-world currency; but with Project Wonderland still in development (current version 0.5), the chances of improvement are high. With respect to our concept for object security, Project Wonderland provides an ideal experimental environment.

4 CONCEPT FOR OBJECT SECURITY

Virtual worlds have been around for some time, yet the research centering on object security in multi-environment conditions is still quite undeveloped. As we noted earlier in the chapter, securing objects in single environments is mainly done through restricting access by authentication and access control lists, while the multi-environment focus on the transfer of objects is mainly centered on standards discussions; cf. Hurlimann (2009) and OpenSimulator (2010b). Industry to date is interested more in protecting music and movies, instead of 3D objects in virtual worlds. Numerous publications and research streams cover watermarking, encryption, and signatures, but almost none focus on 3D object security in multi-environments. More needs to be done on how we can guarantee that objects are unique or limited to a certain count. Note that we are aware of the influence of market shares and revenues involved in the entertainment industry for music and movies versus 3D-objects in virtual worlds.

Moreover our concept may not shake the view of the (virtual) worlds because we cover known technologies from other fields and existing environments, but we integrate them into a trusted network of objects and certificated environments. Our research argues that people need to be aware of security, and be aware that playing in and learning about virtual worlds has to mesh with the business needs for serious applications. We should also emphasize that our research does not move away from the concepts and philosophy of open objects, but many users are hesitant to fully use virtual worlds because they do not want to lose their ideas and properties. We realize there needs to be a balance between freedom and security.

Our proposed architecture uses technology already implemented within the Internet, for data protection, or digital right management to manage objects in multi-environments. Our concept surrounding the protection of objects includes encryption, certificates, safe and trusted environments, trusted networks, and therefore participation of environment providers to implement this support and protection. In terms of our ongoing research, we have initiated the implementation of the *Global Object Management System* and currently conducting our first experiments with mobile applications. Our concept of securing objects demonstrates that we are able to protect our work, transfer it to other environments, and by this improve their overall value of objects. In addition, secure objects will create new marketplaces as companies can control the penetration of the market by limiting objects (e.g., collectibles with increased value) or prevent plagiarism and manipulation, thereby protecting copyright and intellectual property.

Our architecture combines several ideas common to digital goods controls on the Internet (images, movies, books, or music), as well as virtual worlds (OpenSimulator, Project Wonderland), in conjunction with encryption and security concepts (PGP, watermarks, DRM). The following sections describe our concept and some implementations of GOMS, how it can be integrated in virtual environments, and what current research questions on which we are currently focusing. With

respect to underlying technologies, we refer to a collection of books and papers for further reading at the end of our chapter.

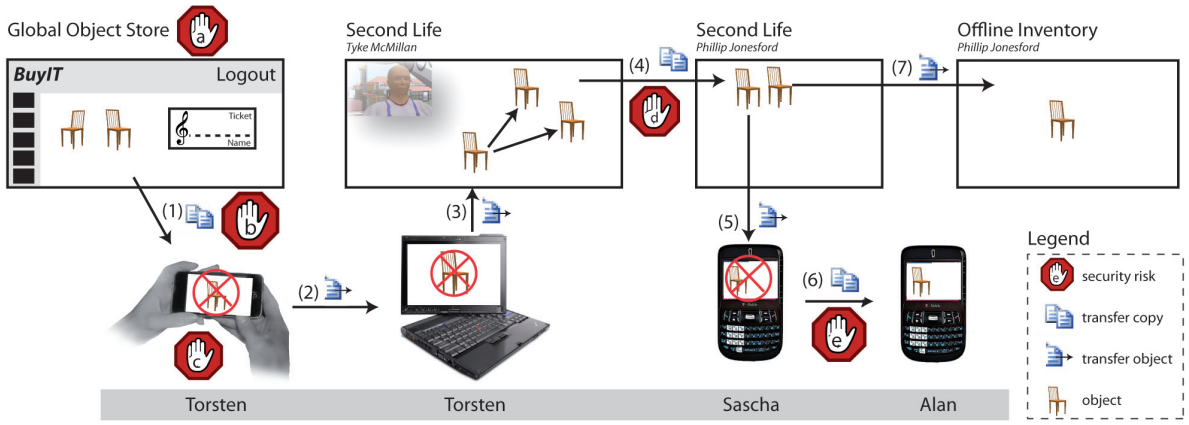
4.1 Global Object Management System Architecture

This section outlines our system architecture created to handle digital objects and provide (if requested and anticipated by the object) a secure exchange between different environments. We discuss it in this section realizing that the presented system is reduced to elementary components to visualize the idea, while keeping the complexity low (cf. Section 4.2 and Figure 14).

Figure 6 shows a possible exchange of objects passing through different devices and platforms. In this example, Torsten buys the object *chair* (originally created by Torsten's Second Life avatar Tyke) and keeps the copy on a mobile device (1). For this example, we assume a global limit of four objects *chair*; including the original in the store. With respect to the environments, we assume the blackberry to be certified and secure, while the iPhone and Notebook are not certified. Torsten has no privileges besides transferring the object to other devices (2) or environments (3). In *Second Life*, Tyke (the next owner of the object) is allowed to create copies (up to a given limit) and passes two newly created *chairs* to Phillip (4) in the same environment. Phillip keeps one version in his personal inventory (7, transfer) and transfers one version to the mobile device of his (real-life) user Sascha (5, transfer); with no privileges for Sascha to do any modifications to the object. Later, Sascha intends to give another copy of the object to his colleague Alan as a present (6). Unfortunately, we have created two extra copies of the object *chair* since the first transfer (in total, there are already four *chairs* in the system) and reached the upper boundary; preventing us from actually giving Alan a copy of the *chair*. Note that Alan's mobile device is a secure environment and, therefore, objects are verified and checked against limits thereby stopping the transfer. However, on unsecure devices, this might not be checked, but the object would be protected by encryption, thereby stopping the transfer as well; cf. Section 4.1.3 and 4.1.4. In conclusion, we have to deal, for example, with unsecure (in the following called uncertified) environments, change of ownership, checking the availability of objects, and securing the transfer.

Figure 6 also includes transactions that may result in possible GOMS security risks: These include (a) authentication, (b) secure transfer of

Figure 6. Example for the turnaround of an object passing through several certified and non-certified systems and environments.



objects, (c) protection of objects on non-trusted devices, (d) multiplying and transferring objects, and (e) guaranteeing the given limit of existing objects in the system. Our example demonstrates the need for GOMS to control security of the object and its properties (e.g., scripts or metadata), keep track of objects (e.g., ownership, count, or location), entities (users, avatars, environments), and regulate processes (transfer, copy, filter, transformation). By using strict authentication, key-pair encryption, hash codes, and certificates for environments, we build a network of trusted components, in which object security can be controlled and guaranteed to a certain extent. This is similar to the trusted computing framework and its strong hierarchical approach to secure systems all the way from hardware to applications, which guarantees non-manipulated systems (Trusted Computing Group, 2010).

In case of missing privileges and certificates, the encrypted objects are not usable on, for example, Torstens' mobile devices (non-trusted system), even though the devices can pass the digital files on to other users and avatars. Alan, on the other hand, owns software which is trusted and able to decode objects for visualization if his device has the appropriate object permissions and access. For the sake of completeness, we call objects **invalid** if the object is modified and not yet verified for correctness by GOMS or transferred from an uncertified environment without being completely encrypted. Otherwise, objects are considered to be **valid**. The object is valid as long as it is only in certified environments, whereas GOMS is required to verify availability, which cannot be done otherwise; cf. Section 4.1.4.

Multiple ownership of objects does not imply that all owners have their individual copy, considering that the same object cannot exist more than one time; for example, a unique piece of artwork. It merely means that the object should be simultaneously accessible by all owners. In this situation, GOMS keeps track of transfers, knows about the current owner (of the *physical* object) and location (device), and supports communication and exchange of objects. In addition, GOMS acts as a global object server providing access to replicas of objects if, local repositories or inventories are offline. Figure 14 in Section 4.2 shows the major components and processes of GOMS, and the following subsections (4.1.1-3) describe details about how GOMS handles potential security risks and required processes.

4.1.1 Authentication

Granting access to objects, servers, environments, or functionalities can be implemented in many ways. In this case we need to control the access to system components (e.g., accessing a store), allowed functionality within the components (e.g., browsing the catalog, buying objects, (unrestricted) transfer to other systems), or the privileges being granted for specific objects. *Access control lists* (ACL) are the common solution to control access. Similar to the UNIX file system, ACL govern distinct access on different levels (*read, write, execute*) for different user groups (*owner, group, everyone*). Whereas the level of detail can vary in other systems, here access is granted to identities. For example, Second Life uses *creator* and the permission to *transfer* or *copy* objects, whereas visibility (read) is not realized.

For the sake of completeness, we mention other concepts but will not go into further details because we use privileges to encode access to objects. The other concepts are *role based access control,* (RBAC, users or groups have assigned roles with specific access to resources), *discretionary access control* (DAC, restricting the access to objects for users or groups, similar to Unix models [cf. Wright & Madey, 2008a]), *mandatory access control* (MAC, also called non-discretionary access control, which uses multi-level security), *capability-based security* (CBS, in which the user needs capabilities to access or interact with referenced objects according to the rights of the capability, [cf. Scheffler, 2008]), and *rule-based*

access control (RuBAC, *a* set of rules that defines the access to resources [cf. Techtopia, 2009]) for further information about this subject.

In addition to different models for controlling the access, we need to specify the kind of authentication, such as providing proof that one is the authorized individual. Basically, the spectrum varies from single factor authentication of username-password with different exigencies for the password quality up to multi-factor authentication with combined biometric identification, password authentication, and device credentialing (Vielhauer, 2005). With respect to object security, we will later describe key-pair encryption with secret and public keys. If a message or object is encrypted with the public key, it can only decoded by the matching secret key, therefore identity is proven by owning the correct key (in addition to keeping the secret key, the validity of the public key is also verified by trusted networks [Haenni and Jonczy, 2007]). On the other hand, the secret key can be used to sign messages (encryption), with the public key as the matching counterpart for decryption or identity check, respectively (cf. Ferguson et al. (2010) for more details on key pair encryption).

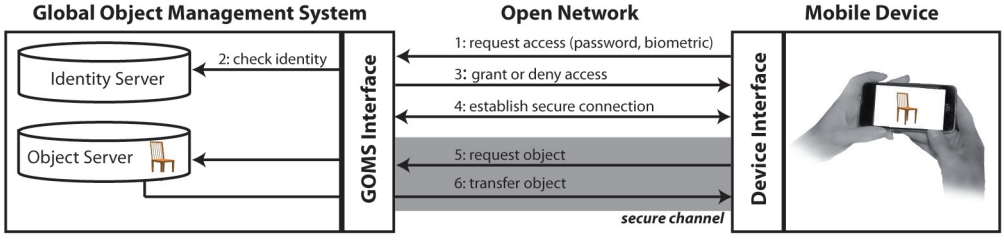
4.1.2 Encryption

In case of secure environments (e.g., object stores, mobile devices, virtual worlds), the interception of objects during transfer is the most likely attack vector against content. Secure connections (e.g. virtual private networks (VPN) or secure protocols like SSL) and authentication of communication partners are critical protections to prevent attacks (e.g., man-in-the-middle). In addition, the objects require encryption in case of illegal acquisition so that the objects are of no value for the attacker since they are undecipherable (cf. Cole (2009) for an overview for network security). Figure 7 depicts the processes for authentication and secure transfer in open networks. In this example, the user requests access via a mobile device using login and password (1), which is authenticated by the identity server (2). Furthermore, a secure channel is established (4) to transfer (6) the requested object (5). Note that this example excludes the GOMS encryption visualization and focuses on the authentication and transfer.

Advanced protection is required when objects are in open (non-certified and non-trusted) environments; e.g. Torstens' mobile device in Figure 6. In this is situation, object access, in conjunction with any modifications, duplications, or usages, needs to be prevented. Objects have properties and content with discerned access levels that need to be maintained. In many cases, it might be expedient to allow readability of properties for retrieval or classification. Nevertheless, the integrity of the object has the highest priority and requires several mechanisms that must be included in the security model.

Depending on the kind of object (free vs. payment, open code vs. hidden code), the content may or may not be visible. One option would be the inclusion of previews while the full content is kept encrypted without the required matching

Figure 7. Authentication and secure transfer of objects; encryption is not shown here

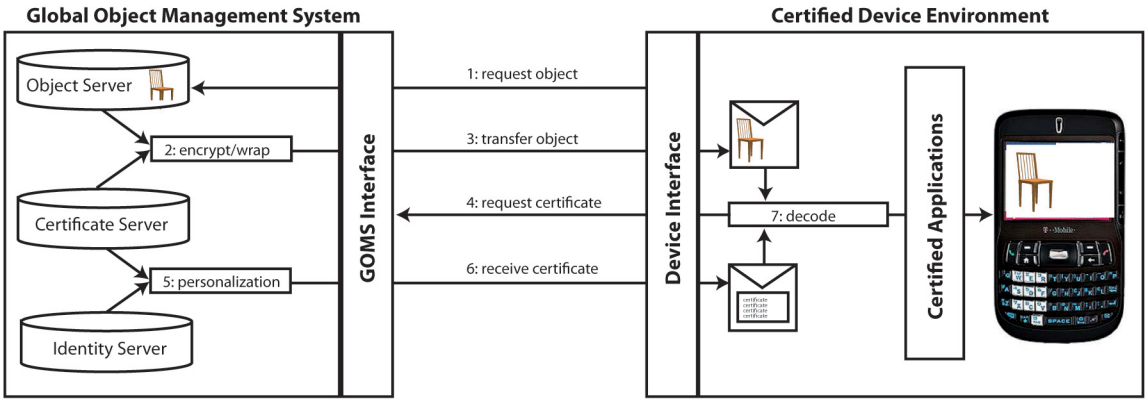


keys or certificates. Previews could also censor adult themes until required authentication had occurred. Integrity is assured by encrypted hash codes, which are inserted by GOMS or respective environments using their secret keys. Note that in open (uncertified) environments encryption can, and perhaps should, be used on the whole object (to include properties and content) in order to prevent any access. The object would be stored on the device without further application or usage. One scenario to use this approach would be the backup on (public) storage places to distribute the risk of data loss. Further description on hash codes is discussed below within the context of object transfer (cf. Section 4.1.3 covering object transfer).

Another context for objects is a secure, certified, and trusted environment. Here, the content is (in certain boundaries) protected against external risks and can be decrypted by entities with sufficient privileges. Figure 8 demonstrates the practice of certificates where requested objects (1) are encrypted and wrapped in envelopes (2) before being transferred to the requesting environment. Not shown in Figure 8 is the identity authentication illustrated in Figure 7, with which the enveloped is addressed and sealed. Just as with public key encryption, the envelope can only be opened with a matching certificate. The corresponding certificate (matching the previously encoded object) is requested (4), personalized (5), and sent back (6) to open the envelope (7). The personalization includes further details on what the identity is permitted to do. In our model this concerns access to properties and content as privileges (regarding the object) are part of the object itself. In step (7), it is important that the environment is trusted and verification holds against attacks and security holes. Much of this is accomplished in the system framework as trusted systems require a complete stack of authenticated components, from hardware to the high-level software application. The object can be decoded by opening the envelope with the certificate and decrypting the content. Further regulation of privileges is done by the environment, which guarantees the object security and blocking of attacks from internal and external threats and unauthorized access.

In general, the current active environment is responsible for the security of the object before, during, and after the transfer to new environments. This holds true for certified environments that are part of the complete GOMS architecture. Nevertheless, we must consider uncertified environments, as well as two modes of connectivity: online and offline. Being online is straight forward as the transfer involves GOMS, which tracks and maintains identities, certificates, object states, and

Figure 8. Using certificate to protect objects from being used in environments other than certified ones



availability. Therefore, the object can be transferred via GOMS to perform all checks (cf. Section 4.1.3). However, offline transactions require further precautions. If objects are moved to uncertified environments, they are wrapped with an envelope using the public environment key (issued by GOMS during certification). The following actions would be performed for increased object security:

- **Content Encryption:** The environment creates a symmetric key to encrypt the content. The key itself is stored (with the object) and encrypted for each identity having access rights (defined by the privileges) with the corresponding public key. This encryption allows decryption by different entities without distributing the actual key; c.f. Figure 9.
- **Property Encryption**: Encryption similar to the key for content, but designed either for all properties as a whole or each individual property to provide selective access control. This key can be valuable if one wants to keep general information for the object accessible for search engines while other properties are kept secret.
- **Integrity**: Maintaining an object's desired state and behavior is most important in uncertified environments to protect against modifications, as well as achieve access in secure environments. Encrypted hash codes provide fast verification against unauthorized modifications by recalculating and comparing the hash code to the decrypted matching public key hash code. Deviations result in invalid objects.

Modification should also be considered using hash codes to assure integrity. If online, each modification can be verified by GOMS (including privileges) which will create and sign new hash codes. Offline hampers the process, but with change and revision tracking in place (including prove of identity), GOMS can use this information to later verify, approve, and assign new and valid hash codes to the object.

We illustrate this procedure in Figure 10 as follows: (1) Calculate hash codes for different parts of the object, thereby identifying the corresponding hash code (for which modifications are anticipated). (2) Re-calculate the current hash code (HC_{TR1}). (3) Copy the original content to the trace for later verification, reconstruction, and protocol of modification. (4) Perform the modifications (here *Stuhl* to *Chair*). (5) Calculate the new hash code (HC_{TR2}). (6) Sign both (old and new) hash codes with the private key such that GOMS can verify the identity by using the matching public key. (7) Add the signed hash codes to the list of hash codes. (The list contains the original hash code and two for each modification).

Figure 9. Access to encrypted content by multiple entities using encryption of onetime keys

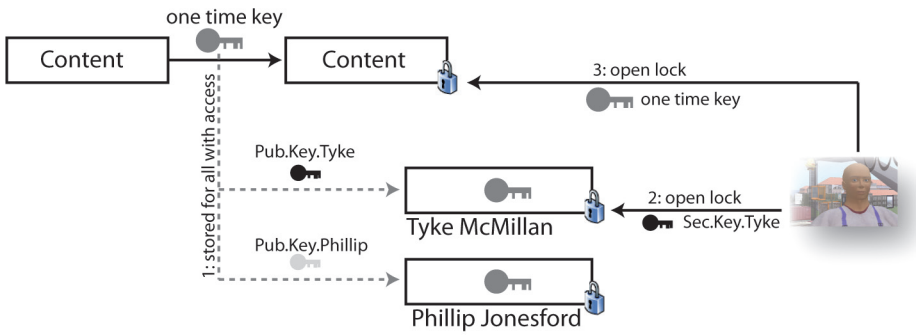
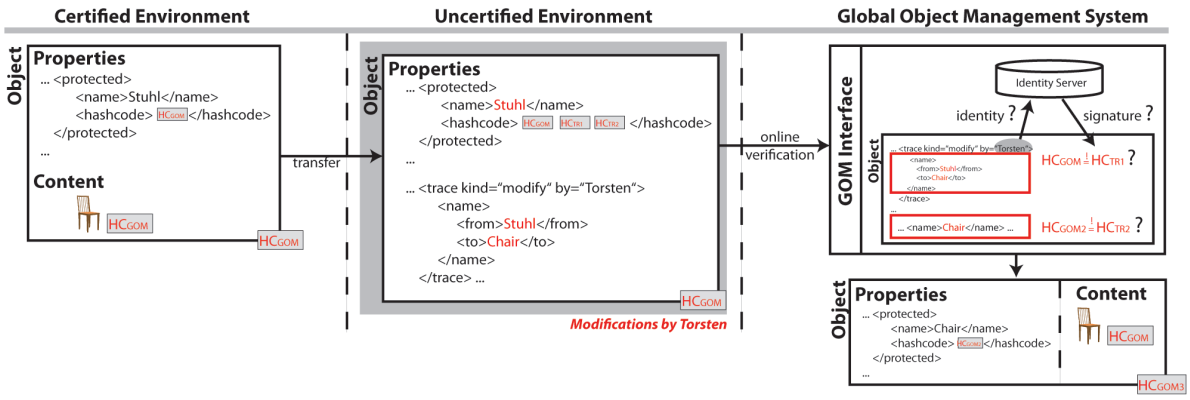


Figure 10. Protection of integrity if accessible properties are modified in uncertified environments. The approval is done by GOMS. Here, the hash codes are given for the name, content, and overall object



GOMS completes the online verification as well. Here, the signed hash codes (calculated and signed by the identity who did the modifications) are compared to the ones from GOMS (HC_{TR1} with HC_{GOM} and HC_{TR2} with HC_{GOM2}). If the hash codes do not match, the object is considered invalid (assuming that someone did [unauthorized] modifications). Otherwise, the modification is accepted and the object is compressed by removing hash codes and recalculating the overall object hash code once again.

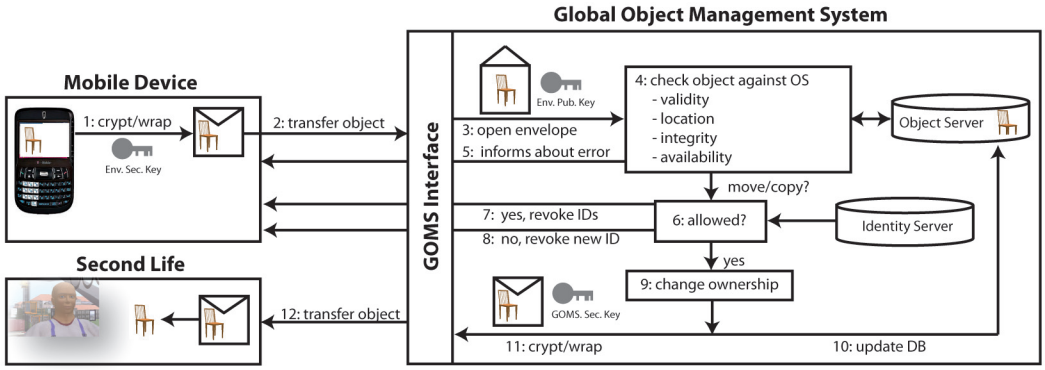
Object are deemed invalid if re-calculated hash codes do not match the signed hash code in the object, or if there is a list of hash codes for any part (implying that modifications of the object were not verified; cf. Section 4.1.3) Although certified environments might be able to provide functionality for integrity verification, they cannot control limitations on the count of objects unless being connected to GOMS (cf. Section 4.1.4).

4.1.3 Transfer of Objects

Without distinguishing between copy and move, there are eight kinds of transfer in GOMS, that are either offline or online, as well as all combinations of certified and uncertified environments. As discussed previously in the chapter, transfer between uncertified environments cannot be controlled in either connectivity mode. All that can be assured is to protect the content through encryption and rely on later GOMS verification. Transferring objects from certified to uncertified environments while being offline can be done by encrypting required parts (content), guaranteeing integrity through hash codes (mainly properties), and wrapping the object in an envelope (requiring a certificate or matching key to open the envelope again).

Figure 11 depicts the online transfer of objects from certified to other environments via GOMS. The wrapped object (1, using the private key of the environment) is transferred to GOMS (2) where it is first unwrapped (3) and checked against several criteria (4). GOMS serves as storage facility for all objects including information about their current location, quantity, and ownership. GOMS also verifies if objects are valid and have not been manipulated by unauthorized entities. Therefore, each incoming object is checked for its expected **location** (problems might result from transfers to non-certified environments), its **validity** (certified environments can invalidate objects if they are deleted or show manipulations), **integrity** (hash codes have to match the content and properties), and **quantity** (does a copy exceed the maximum

Figure 11. Process of verifying the transferred object



number of globally available objects). The checks also include verifying and processing the modifications still being encoded in the objects (cf. 4.1.2).

If checks fail, the transfer is cancelled and the initiating environment is informed to revoke the copied object or inform the user (5). Problems with the object are then handled by the environment itself or the object is send to GOMS for repair. If the transfer is allowed (6, the object can be copied or moved, including the privileges of involved identities), the source as well as new object id in the originating environment is revoked (7, object is invalid), otherwise the transfer is cancelled and the id of the new object is revoked (8). Having the environment revoke objects is necessary because moving and copying objects can leave objects in the cache, and the objects need to be declared as invalid (responsibility of the environment, whereas GOMS would not further issue certificates for these objects). The ownership is modified and the privileges are updated. Finally, the object is encrypted, wrapped (11), and sent to the destination environment, which could be the same as the originating one (12). In addition, the object database is updated according to the object properties and content (10). Information about the transfer is associated with the object for later traceability, with information such as environments, type of transfer, involved identities, and changed ownership.

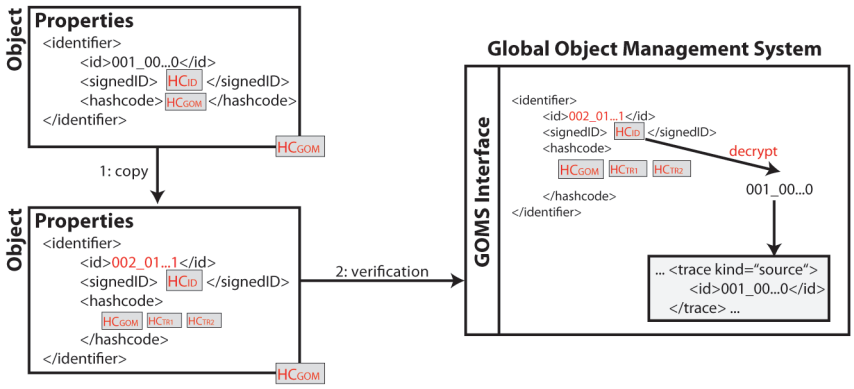
Copied objects reference their source via the identifier in the trace; cf. also Figure 4. This ensures that the source identifier of a copied object is not removed (loss of all references). The identifier is also duplicated in the object with the second identifier being signed by GOMS as with the hash codes. As a result, GOMS is able to recognize copies and their origin. During the checks (cf. Figure 11), the trace is updated according to the signed source identifier (cf. Figure 12).

4.1.4 Uniqueness of Object Identifier and Limited Quantity

Our proposed architecture combines two philosophies: global server-based management and simultaneous self-sustaining objects. We have chosen this approach to strike a balance between handling and security. The previous sections provided an overview about security and transfer procedures, but did not discuss the concept of securing uniqueness and boundaries on limited objects. Again, the main concern is that objects can originate from uncertified environments where online-connection to GOMS is not guaranteed at all times. Therefore, new objects are considered to be invalid as long as they are not verified and approved by GOMS.

If we assume that an object is created in an uncertified system (e.g., XML-editor), it is not

Figure 12. Securing integrity of identifier and inheritance tree during copy

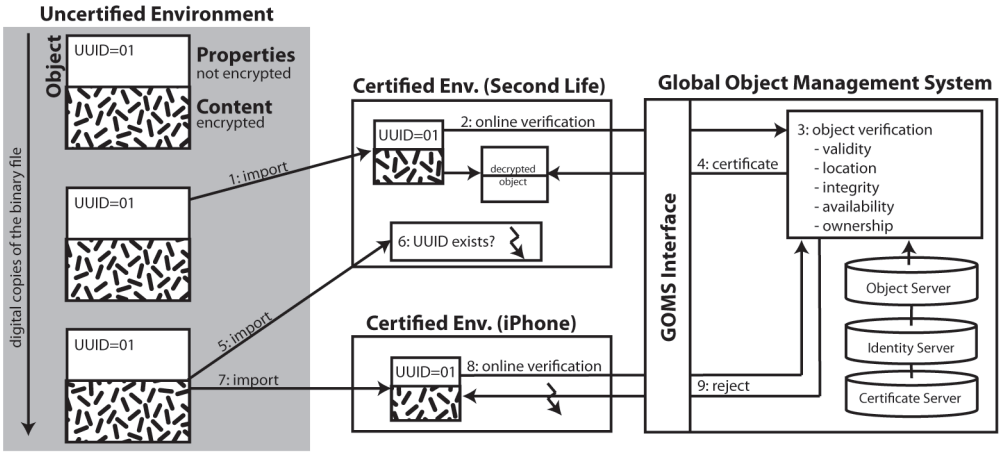


possible to define unique identifiers without contacting GOMS; thus the creator program might define its own identifier. The identifier is flagged as temporary (either by parameter or the missing signed hash code [by GOMS]) and has to be later adjusted by GOMS (e.g., using an [Web] service or importing it into a certified environment). While the object is part of GOMS, the object server keeps track of the unique identifier such that it is not re-assigned as long as the object is still used. The identifier can be re-used if objects are permanently removed.

In addition to managing the uniqueness of objects, GOMS controls limits on objects. First, the usage of objects relies on certificates (certified environments) and strict encryption of critical parts in uncertified environments. Second, the validity and integrity of objects is approved only by GOMS (by hash codes), which is also keeping track of all objects and their count. Thus, GOMS holds limits by not validating new objects or by restricting issued certificates. Let's have a look at another example: Protection against stealing. When objects leave a certified environment, they require proper protection by having them *sealed* (signed) and/or encrypted with the corresponding secret key of the owner. Protection against modification is given by hash codes while stealing is prevented by (1) encryption and (2) the challenge of seal removal from the original owner. This is especially true if copied and passed along to someone else within uncertified environments. Here, the transfer has to be later authorized by GOMS involving authentication by both (new and old) owners.

Figure 13 depicts the verification process from a higher system level. Here, if an object is copied several times within an uncertified environment, the verification during import will either result in acceptance or rejection of the object. The verification of ownership is not shown in Figure 13, but is part of Step 3. In general, the imported object (1, 5, and 7) is first verified by GOMS (2 and 8; requiring an online connection) regarding the different criteria (3): validity, expected location (here, objects should be reported to have left a certified environments before or the last know environment is contacted to verify the transfer), integrity, ownership, and availability. The outcome of the verification results either in decrypting the object by passing a certificate to the environment (4) or rejecting the object. Most important in this context is the UUID, the ownership, the trace of the object to verify transfers, and signatures as well as hash codes (cf. our discussion in the previous sections).

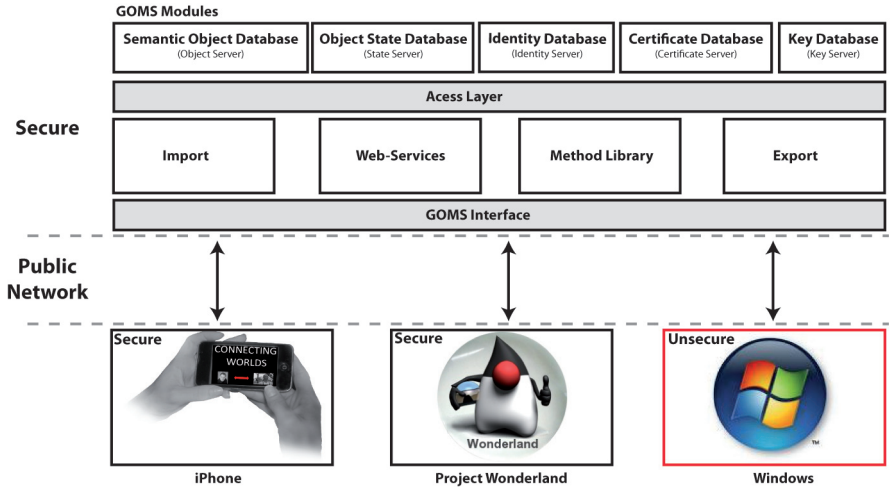
Figure 13. Import of (non-approved) copies into certified environments



4.2 GOMS Architecture in an On-Going Research Project

GOMS is research in progress. Thus, several components exist mere as experimental prototypes rather than releasable modules. But the implementations so far demonstrate that we are pursuing a promising research stream with the architecture and able to provide a secure environment for critical projects in an emerging field. All of the previously described concepts are building blocks for GOMS, with the overall schematic illustrated in Figure 14. There are additional modules here not discussed in this chapter because they do not directly deal with object security management. These modules are primarily the filters (import, export, conversion) and the databases for building the (semantic) network of objects. The following list presents a short overview and description of the modules:

Figure 14. Modules of the Global Object Management System and currently applied test environments in the ongoing research project



- **Environments**: We focus currently on three environments, one of which is simply the uncertified Windows file system to verify the security of objects. The others are Project Wonderland, where we have the highest degree of flexibility for experiments in virtual worlds and the IPhone with a self-developed mobile store for 3D objects. We consider the last two environments to be secure and trusted. OpenSim is a further candidate currently being considered to expand the number of environments.
- **Databases**: GOMS requires storage of objects, certificates, and identities. Here, we use different database models with respect to the stored data. The core component is the development of a **Semantic Object Database**, which does not follow the relational database model but builds networks representing the relations of the objects (cf. Aust and Till, 2009). This database stores the actual object and uses the information in the properties (and especially relations) to build a network of objects. In combination with *intelligent* import filters to analyze tags with respect to concepts (including WordNet), we are able to build automatic relations and increase the value of the network (cf. Boese, 2010). Other databases we use are the **Object State Database** (information about identifier, validity, location, and ownership), the **Identity Database** (information about all entities including groups), the **Certificate Database**, and **Key Server**. The object and object state databases are separated for performance reasons, whereas the second one contains operative data for repetitive request, while the first is rather seen as a data warehouse and advanced requests. Note that we consider database and server to be synonymous as the databases include technology to access them via network and to process requests in a more advanced way than just returning the results.
- **Export and Import**: Controlling all incoming objects and requests including the described verification on validity (cf. Section 4.1.3) and converting (if possible) of objects during export to a specific environment. Here, it might be important to emphasize again, that compatibility in multi-environments is as problematic as security. We use an object model, where the multitude is encompassed but limits still exist especially with respect to functionality. If conversion is not possible, export might be denied. The filtering and conversion process is not further described here as this is only of minor relevance for security.
- **Web-Services**: Most of the functionality should be accessible by other applications in their simplest form. Precisely, GOMS provides services to verify the validity and integrity of objects, to retrieve objects from the (semantic) network, convert objects, and other functions.
- **Interfaces**: Communication to environments and applications is established through the GOMS Interface (also responsible for establishing a secure communication channel). The databases are accessed via an Access Layer controlling requests for data; if more than one database is involved.

The above modules were not discussed in this chapter as they do not immediately apply to object security. Although they are critical for smooth GOMS operation, they are not application here. Additional discussions of the GOMS architecture as it relates to object security within e-learning (Reiners et al., 2005; Reiners & Suhl, 2007) are available. With respect to our ongoing research, we are currently implementing the next version of the encryption model for a larger experimental

setting. We do want to note that an initial experiment using the iPhone produced promising results that confirm the malleability and functionality of the GOMS architecture to adapt to various media while controlling the flow of objects. More about the GOMS architecture and our research is available at our website (http://www.virtual-object-security.org).

5 CONCLUSION

Based on the Gartner Hype Cycle, (public) virtual worlds are far beyond their *peak of inflated expectations* and expected to reach the *slope of enlightenment* within the next years with some fields like education already well beyond this point. During the gold-rush atmosphere the speed of development resulted in numerous virtual environments and uncountable amounts of objects with little concern about security for the objects, but finding (economical) applications. However, it seems that with the revival of 3D entertainment and the shift to a 3D Web, where virtual worlds and Web 2.0 define the common denominator, it is more or less a matter of time. Add to this the importance of security not only of the environment and avatar (through authentication) but also the objects, which are the essential building blocks for a dynamic and living environment.

In this chapter, we proposed a different approach for encoding virtual objects following the object oriented programming paradigm. All information is stored within the object, such as privileges for access and behavior functionality. We illustrated contemporary approaches, such as in Second Life to demonstrate the comprehensiveness of current and coming applications, always keeping in mind the need of object security and exchange between different environments. In addition, we sketched our GOMS architecture consisting of trusted environments and interconnected servers to guarantee object security. We picked major processes to demonstrate the object handling with transfer, integrity, and availability. Key components of GOMS are the encryption, usage of hash codes, and wrapping in sealed envelopes requiring certificates to be opened again. This approach follows accepted security standards currently used in digital right management implemented, with arguable success, in the entertainment sector. We are aware that our approach involves obstacles and should be used with care in real-life scenarios. Nevertheless, security for objects must happen in various contexts so that an overlay of encryption and trusted environments is accepted as standard procedure rather than the alternative of unauthorized copies or access.

GOMS is ongoing research and its implementation in process. Select components exist and function well. This encourages us to continue on our path. The demand for security exists and we propose our solution to redefine the object and its handling. Coming from an educational background, we are concerned with (maybe minor) cases from creating objects in classroom or exams up to innovative product development in distributed virtual teams, where, for example, prototypes of new engine concepts (in form of 3D objects) are exchanged via the Internet and could be illegally acquired. However, in business, a breach of object security would be more dire. The loss of intellectual property or trade secrets could cause an economic catastrophe if the invention becomes public before securing, e.g., patents. GOMS is a systematic approach to guard against this occurring so that organizations can innovate in virtual realms knowing that intellectual property encapsulated in objects will be protected by the very system they use to create objects.

6 ADDITIONAL READING

Numerous publications and research streams cover watermarking, encryption, and signatures, but almost none focus on 3D object security in

multi-environments. More needs to be done on how we can guarantee that objects are unique or limited to a certain count. We mentioned and used various technologies, which are important for the global object management system but beyond the scope of the paper. We suggest the following sources as a starting point to dive deeper into the subject (of which some were already cited in the chapter): Encryption in general (Ferguson et al., 2010; Schneier, 2007; Furht et al., 2005), Pretty Good Privacy (PGP; Garfinkel, S.), network security (Cole, 2009; Ciampa, 2009), Digital Right Management (Torres Padrosa, 2009; Zeng et al., 2006), digital watermarking (Eshraghi & Samavati,, 2007; Ho et al., 2009; Cox et al., 2007).

Virtual worlds go e-learning. The major application for virtual worlds is education, i.e. distance/blended learning or classes requiring a certain amount of travel (geography, analyzing different cultures), bringing alive history (visiting sites, rebuilding the past), visualize scientific experiments (physical settings for gravity, teaching astronomy), or interacting in role plays (theatre, discussions, psychology). The number of papers is endless; some examples (among our very own publications) are Dreher et al. (2009a), Dreher et al. (2009b), Reiners et al. (2009), Gregory et al. (2009), Daniels Lee (2009), and Jäger & Helgheim (2009)). But virtual worlds spread to almost every field, some resources (without referencing further individual publications) are the Journal of Virtual Worlds Research (http://jvwresearch.org), Virtual Worlds News (http://www.virtualworldsnews.com), the reports from KZero (http://www.kzero.co.uk), and the blogs of and about virtual worlds or their (e.g., Second Life at http://blogs.secondlife.com and http://nwn.blogs.com). For new ideas, even novels can provide some background (Stephenson, 1994). Last, we should have a short look at virtual worlds and their objects. Especially, as virtual objects are getting valuable and sell for over $330.000 USD (Jones, 2010); see also Muttik (2008), Sastry (2007), and Lee (2009).

7 REFERENCES

Aust, T., & Sarnow, M. (2009). *Entwurf und Implementierung einer Medien-Datenbank-Middleware mit integrierten Semantischen Netzen* (engl.: Design and Implementation of a Media-Database-Middleware with Integrated Semantic Networks). Diploma Thesis.

Baca, M. (2008). *Introduction to Metadata*. Getty Research Institute.

Berners-Lee, T., Fielding, R., & Masinter, L. (2005). *Uniform Resource Identifier (URI): Generic Syntax*. Retrieved May 15, 2010, from http://tools.ietf.org/html/rfc3986

Blizzard Entertainment. (2010). *World of Warcraft Community Site*. Retrieved May 15, 2010, from http://www.worldofwarcraft.com

Boese, S. (2010). *Entwicklung und Umsetzung einer konzeptualisierten Speicherung von Dokumenten in einer semantischen Datenbank* (engl.: Development and Implementation of a Conceptualized Storage of Documents in a Semantic Database). Diploma Thesis.

Castronova, E. (2003). *The Price of 'Man' and 'Woman': A Hedonic Pricing Model of Avatar Attributes in a Synthethic World*. CESifo Working Paper Series No. 957. Retrieved May 15, 2010, from http://ssrn.com/abstract=415043

Ciampa, M. (2009). *Security+ Guide to Network Security Fundamentals*. Course Technology.

Cole, E. (2009). *Network Security Bible* (2nd ed.). Chichester, UK: Wiley.

COLLADA. (2010). *COLLADA: Digital Asset and FX Exchange Schema*. Retrieved May 15, 2010, from http://collada.org

Cox, I., Miller, M., Bloom, J., Fridrich, J., & Kalker, T. (2007). *Digital Watermarking and Steganography* (2nd ed.). San Francisco: Morgan Kaufmann.

Daniels Lee, P. (2009). Using Second Life to Teach Operations Management. *Journal of Virtual Worlds Research*, *2*(1), 3–15.

Dreher, C., Reiners, T., Dreher, N., & Dreher, H. (2009a). 3D Virtual Worlds as Collaborative Communities Enriching Human Endeavours: Innovative Applications in e-Learning. In *Proceedings of IEEE DEST, TD-001252* (2009).

Dreher, C., Reiners, T., Dreher, N., & Dreher, H. (2009b). Virtual Worlds as a Context Suited for Information Systems Education: Discussion of Pedagogical Experience and Curriculum Design With Reference to Second Life. *Journal of Information Systems Education, Special Issue*, *20*(2), 211–224.

Ducheneaut, N., Wen, M., Yee, N., & Wadley, G. (2009). Body and mind: A study of avatar personalization in three virtual worlds. In *27th Annual CHI Conference on Human Factors in Computing Systems (CHI 2009)*, April 4-9; Boston, MA.

Erlenkötter, A., Kühnlenz, C.-M., Miu, H.-R., Sommer, F., & Reiners, T. (2008). Enhancing the class curriculum with virtual world use cases for production and logistics. In *Proceedings of E-Learn 2008: World Conference on E-Learning in Corporate, Government, Healthcare, & Higher Education,* Las Vegas, (pp. 789–798).

Eshraghi, M., & Samavati, F. F. (2007). 3D watermarking robust to accessible attacks. In *Proceedings of the First International Conference on Immersive Telecommunications* (Bussolengo, Verona, Italy, October 10 - 12, 2007). Brussels, Belgium: ICST (Institute for Computer Sciences Social-Informatics and Telecommunications Engineering).

Ferguson, N., Schneier, B., & Kohno, T. (2010). *Cryptography Engineering: Design Principles and Practical Applications*. Chichester, UK: Wiley.

Furht, B., Muharemaqic, E., & Socek, D. (2005). *Multimedia Encryption and Watermarking*. Berlin: Springer.

Garfinkel, S. (1994). *PGP: Pretty Good Privacy*. Sebastopol, CA: O Reillly.

Google Code Labs. (2010). *O3D API: Google Code*. Retrieved May 15, 2010, from http://code.google.com/apis/o3d

Gregory, S., Reiners, T., & Tynan, B. (2009). Alternative realities: Immersive learning for and with students. In Song, H. (Ed.), *Distance Learning Technology, Current Instruction, and the Future of Education: Applications of Today, Practices of Tomorrow* (pp. 245–272). Hershey, PA: IGI Global.

Gütl, C., Chang, V., Kopeinik, S., & Williams, R. (2009). 3D Virtual Worlds as a Tool for Collaborative Learning Settings in Geographically Dispersed Environments. In *Conference ICL2009,* September 23 -25, Villach, Austria

Habbo Hotel. (2010). *Habbo Hotel US: Make friends, join the fun, get noticed!* Retrieved May 15, 2010, from http://www.habbo.com

Haenni, R., & Jonczy, J. (2007). A New Approach to PGP'sWeb of Trust. In *EEMA'07, European e-Identity Conference*. Retrieved May 15, 2010, from www.iam.unibe.ch/~run/download.php?pdf=HJ07

Ho, A. T. S., Shi, Y. Q., Kim, H. J., & Barni, M. (2009). *Digital Watermarking: 8th International Workshop, IWDW 2009*, Guildford, UK, August 24-26, 2009, (LNCS on Security and Cryptology, 5703). Berlin: Springer.

Horowitz, S. J. (2007). Competing Lockean Claims to Virtual Property. *Harvard Journal of Law & Technology*, *20*(2), 443–458.

Hurliman, J. (2009). *Approaching Virtual World Interoperability*. Retrieved May 15, 2010, from http://software.intel.com/en-us/blogs/2009/02/18/approaching-virtual-world-interoperability

International, D. O. I. Foundation. (2010). *The Digital Object Identifier System*. Retrieved May 15, 2010, from http://www.doi.org

Jäger, B., & Helgheim, B. (2009). Role Play study in a Purchase Management class. In Molka-Danielsen & Deutschmann, M. (Eds.), *Learning and Teaching in the Virtual World of Second Life*. Trondheim, Norway: Tapir Academic Press.

Jones, R. (2010). New Record for most expensive virtual object – $300,000USD. *MetaSecurity: Security of Virtual Worlds*. Retrieved May 15, 2010, from http://metasecurity.net/2010/01/05/new-record-for-most-expensive-virtual-object-300000usd

June, L. (2009). *Amazon remotely deletes Orwell e-books from Kindles, unpersons reportedly unhappy (update)*. Retrieved May 15, 2010, from http://www.engadget.com/2009/07/17/amazon-remotely-deletes-orwell-e-books-from-kindles-unpersons-r

Kabus, P. (2009). *A Network-Agnostic and Cheat-Resistant Framework for Multiplayer Online Games*. Ph.D. Thesis, Technical University of Darmstadt, Germany.

Korolov, M. (2010). *Laid off Wonderland developers to continue project*. Retrieved May 15, 2010, from http://www.hypergridbusiness.com/2010/02/laid-off-wonderland-developers-to-continue-project

KZero. (2009). *Q4 2009 Universe chart: Teens and Adults*. Retrieved May 15, 2010, from http://www.kzero.co.uk/blog/?p=3976

Leach, P., Mealling, M., & Salz, R. (2005). *A Universal Unique Identifier (UUID) URN Namespace*. Retrieved May 15, 2010, from http://www.ietf.org/rfc/rfc4122.txt

Lee, C.Y. (2009). Understanding Security Threats in Virtual Worlds. *AMCIS 2009 Proceedings*. Paper 466.

Linden Lab. (2008). *Open Grid Public Beta begins today*. Retrieved May 15, 2010, from http://blogs.secondlife.com/community/features/blog/2008/07/31/open-grid-public-beta-begins-today

Linden Lab. (2009a). *Extend SL: Open Opportunities for Virtual Worlds*. Retrieved May 15, 2010, from http://develop.secondlife.com/extend-sl/open-grid-protocol

Linden Lab. (2009b). *Next-owner permissions FAQ*. Retrieved May 15, 2010, from http://wiki.secondlife.com/wiki/Next-owner_permissions_FAQ

Linden Lab. (2009c). *Linden Lab Official:About CopyBot, similar tools, and the Terms of Service*. Retrieved May 15, 2010, from http://wiki.secondlife.com/wiki/Linden_Lab_Official:About_CopyBot,_similar_tools,_and_the_Terms_of_Service

Linden Lab. (2010a). *LSL Portal: Second Life WiKi*. Retrieved May 15, 2010, from http://wiki.secondlife.com/wiki/LSL_Portal

Linden Lab. (2010b). *UUID: Second Life Wiki*. Retrieved May 15, 2010, from http://wiki.secondlife.com/wiki/UUID

Linden Lab. (2010c). *Introducing the Second Life Enterprise Beta: Second Life Lives Behind-the-Firewall*. Retrieved May 15, 2010, from http://work.secondlife.com/en-US/products/server

Linden Lab. (2010d). *Linden Lab: Makers of Second Life*. Retrieved May 15, 2010, from http://lindenlab.com

Muttig, I. (2008). *McAffee*. Retrieved May 15, 2010, from http://www.mcafee.com/us/local_content/ white_papers/ threat_center/wp_online_gaming.pdf+virtual+world+object+security

Novak, T. P. (2010). *eLab City: A Platform for Academic Research on Virtual Worlds*. Retrieved May 15, 2010, from http://elabresearch.ucr.edu/blog/uploads/publications/Novak_2010_JVWR.pdf. Submitted to Journal of Virtual Worlds Research Open Wonderland (2010). *Open Wonderland*. Retrieved May 15, 2010, from http://openwonderland.org.

OpenSimulator. (2010a). *Main Page: OpenSim*. Retrieved May 15, 2010, from http://opensimulator.org

OpenSimulator. (2010b). *The OpenSim Hypergrid*. Retrieved May 15, 2010, from http://opensimulator.org/wiki/Hypergrid

OpenSimulator. (2010c). *Scripting Languages: OpenSim*. Retrieved May 15, 2010, from http://opensimulator.org/wiki/Scripting_Languages

OpenTrust. (2010). *Trusted Networks*. Retrieved May 15, 2010, from http://www.opentrust.com/en/market/trusted-networks

Oracle (2010). *Project Wonderland: Go ahead, make a scene*. Retrieved May 15, 2010, from http://labs.oracle.com/spotlight/2008/2008-08-19_project_wonderland.html

Reiners, T., Dreher, C., Büttner, S., Naumann, M., & Visser, L. (2009). Connecting Students by Integrating the 3D Virtual and Real Worlds: We Need 3D Open Source Spaces to Keep Socialization, Communication and Collaboration alive. In T. Bastiaens et al. (Eds.), *Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education 2009* (pp. 3125-3126). Chesapeake, VA: AACE.

Reiners, T., Sassen, I., & Reiß, D. (2005). Ontology-based Retrieval, Authoring, and Networking. In G. Richards (Ed.), *Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education 2005*, (pp. 2349—2354). Chesapeake, VA: AACE.

Reiners, T., & Suhl, L. (2007*). eMANGO: eContentplus EU-Proposal*. Working Paper. University of Hamburg.

Retha, A. (2008). *Notes on Wonderland Cells on the Client and Server*. Retrieved May 15, 2010, from https://elgg.leeds.ac.uk/bms4a2r/weblog/11190.html

Sastry, A. (2008). Security in Virtual Worlds: Blurring the Borders. *Technewsworld*. Retrieved May 15, 2010, from http://www.technewsworld.com/rsstory/59399.html?wlc=1275089297

Scheffler, M., Springer, J. P., & Froehlich, B. (2008). Object-Capability Security in Virtual Environments. *Virtual Reality Conference, IEEE VR '08*, (pp. 51—58). DOI: 10.1109/VR.2008.4480750.

Schneier, B. (2007). *Schneier's Cryptography Classics Library: Applied Cryptography, Secrets and Lies, and Practical Cryptography*. Chichester, UK: Wiley.

Sequeira, L. (2009). *Mechanism of three-dimensional content transfer between the OpenSimulator and Second Life grid*. Master Thesis, Universidade de Trás-os-Montes e Alto Douro.

Slusallek, P. (2010). *XML3D*. Retrieved May 15, 2010, from http://www.xml3d.com

Stephenson, N. (1994). *Snow Crash*. New York: Penguin.

Techtopia. (2009). *Security + Essentials*. Retrieved May 15, 2010, from http://www.techotopia.com/index.php/Security%2B_Essentials

Torres Padrosa, V. (2009). *A DRM Architecture Based on Open Standards: Contribution to an Architecture for Multimedia Information Management and Protection Based on Open Standards*. Berlin: VDM Verlag Dr. Müller.

Trusted Computing Group. (2010). *Home of TCG*. Retrieved May 15, 2010, from http://www.trustedcomputinggroup.org

Vielhauer, C. (2005). *Biometric User Authentication for IT Security: From Fundamentals to Handwriting*. Berlin: Springer.

Vilela, A., Cardoso, M., Martins, D., Santos, A., Moreira, L., Paredes, H., et al. (2010). Privacy challenges and methods for virtual classrooms in Second Life Grid and OpenSimulator. In *2010 Second International Conference on Games and Virtual Worlds for Serious Applications*, (pp. 167—174).

Virtual Worlds Review. (2010). *Virtual Worlds Review*. Retrieved May 15, 2010, from http://www.virtualworldsreview.com/info/categories.shtml

von Lohmann, F. (2010). *All Your Apps Are Belong to Apple: The iPhone Developer Program License Agreement.* Retrieved May 15, 2010, from http://www.eff.org/deeplinks/2010/03/iphone-developer-program-license-agreement-all

Watte, J., & Systems, F. (2009). Virtual World Interoperability: Let Use Cases Drive Design. *Journal of Virtual Worlds Research, 2*(3), 3—13. Retrieved May 15, 2010, from https://journals.tdl.org/jvwr/article/view/727/526

Web 3D Consortium (2010). *X3D for Developers*. Retrieved May 15, 2010, from http://www.web3d.org/x3d

Wikipedia (2010). *Metadata Standards*. Retrieved May 15, 2010, from http://en.wikipedia.org/wiki/Metadata_standards

Winkler, S. E. (2010). Licensing Considerations for OpenSim-Based Virtual Worlds. *Journal of Virtual Worlds Research, 2*(4), 3—16. Retrieved May 15, 2010, from http://journals.tdl.org/jvwr/article/view/871/636

Wright, T. E., & Madey, G. (2008). *Discretionary Access Controls for a Collaborative Virtual Environment*. Technical Report. Retrieved May 15, 2010, from http://www.cse.nd.edu/Reports/2008/TR-2008-12.pdf

Wright, T. E., & Madey, G. (2008). *WonderDAC: An Implementation of Discretionary Access Controls within the Project Wonderland CVE*. Technical Report. Retrieved May 15, 2010, from http://www.cse.nd.edu/Reports/2008/TR-2008-15.pdf

Wright, T. E., & Madey, G. (2008). Discretionary Access Controls for a Collaborative Virtual Environment. *The International Journal of Virtual Reality*, *9*(1), 61–71.

Zeng, W., Yu, H., & Lin, C.-Y. (2006). *Multimedia Security Technologies for Digital Rights Management*. New York: Academic Press.