Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

Virtual ID

Distributed Systems Lab

Dominik Blunschy, Kaspar Etter

bldomini@ethz.ch, etterka@ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Barbara Keller, Jochen Seidel
Prof. Dr. Roger Wattenhofer

July 23, 2012

Acknowledgements

We thank Barbara Keller and Jochen Seidel for supervising this work and, together with Professor Roger Wattenhofer, for giving us the opportunity to write the Distributed Systems Lab on our own project.

Furthermore, a special thank goes to Jan Camenisch, scientist at the IBM Zurich Research Laboratory, for his support in the adaptation of his anonymous credential system to the particular requirements of Virtual ID.

Abstract

Virtual ID is a protocol that constitutes an identity layer for the Internet and a semantic alternative to the World Wide Web. It allows you to prove your identity towards others and to look up attributes of others in a decentralized manner. Being freely extensible with services for all possible purposes, Virtual ID aims to supersede proprietary platforms by establishing a framework of open standards.

In this Distributed Systems Lab, we have written a simplified reference implementation of Virtual ID and an example service that provides anonymous online surveys. Both of these projects include an independent Web interface, where users can manage their virtual identities respectively their surveys.

Contents

Chapter 1

Introduction

1.1 Motivation

Though designed in a completely open and decentralized way, the World Wide Web is increasingly dominated by proprietary platforms, which collect content, products and services from providers and make them available for users. From a technical perspective, however, there is in most cases no reason why such systems have to be controlled by a single company. Nevertheless, only few services in common use today are decentralized. One of the best known examples is e-mail, where users can have different hosts and interoperability is achieved by a common standard (see figure 1.1). In order to apply this architecture to a broad range of services, we need an identity infrastructure to authenticate users across hosts and a semantic protocol to compensate for the lack of intermediary platforms: Virtual ID.

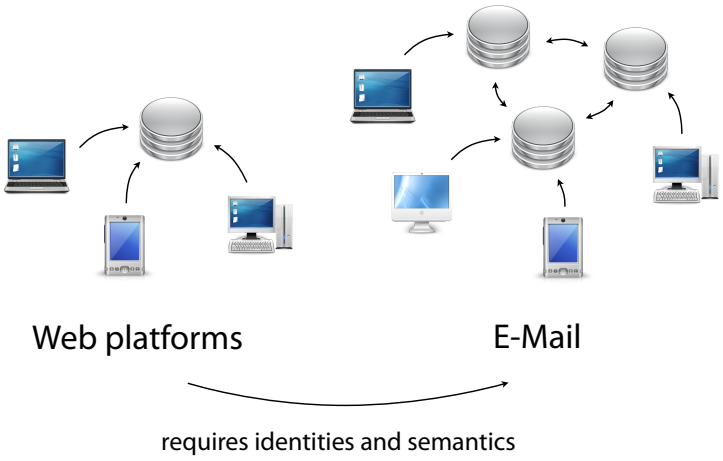

Figure 1.1: Web platforms vs. E-Mail.

1.2 Outline

Our work consists of the following four projects, each of them explained in a separate chapter:

- Virtual ID: Simplified reference implementation of Virtual ID.
- VID Manager: Web interface for users to manage their VID.
- Survey Service: Reference implementation of the Survey Service.
- Survey Manager: Web interface for users to manage their surveys.

1.3 Related Work

Virtual ID with all its concepts is described in the bachelor thesis by Kaspar Etter [1]. Most of its cryptography is based on the anonymous credential system by Jan Camenisch and Anna Lysyanskaya [2].

A good starting point for the study of similar initiatives are the working groups of Identity Commons (www.idcommons.net). One of their most prominent members is OpenID [3], which is a decentralized single sign-on for the World Wide Web. Instead of remembering a different pair of username and password for each platform, users can authenticate themselves with their OpenID. Since requests cannot be signed with an OpenID, however, it is still necessary to have an account at every provider. OpenID is not designed to look up information, but there is an extension to the standard (called the OpenID Attribute Exchange [4]) that facilitates the transfer of user attributes.

A complementary standard to OpenID is OAuth [5], which allows users to grant programs and other platforms access to their resources without providing their passwords. When combined, these three standards cover much of the functionality of Virtual ID. Though the concepts of Virtual ID are more powerful (virtual identities can be merged, roles can be issued to other users, attributes can be certified and shown anonymously, etc.), the two approaches differ not so much in their abilities but rather in their goals: OpenID and OAuth aim to simplify the access to different platforms whereas Virtual ID intends to supersede these platforms altogether.

Chapter 2

Virtual ID

2.1 Concepts

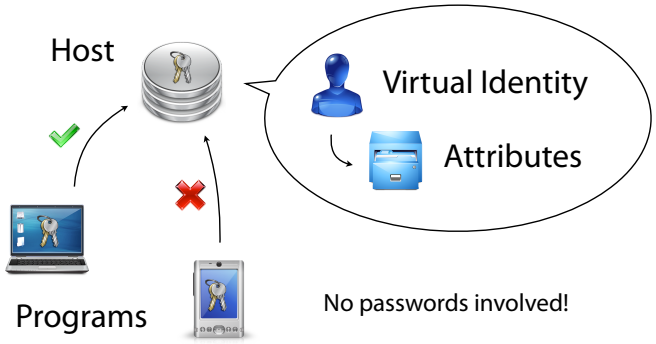

Figure 2.1: The concepts of Virtual ID.

In Virtual ID, natural and artificial persons choose a trusted server to host their *virtual identity* (abbreviated as VID). Every VID has a globally dereferenceable identifier that consists of a user name and the host address combined with an @. Since VIDs can be relocated to new hosts without breaking existing references, several identifiers can map to the same VID. Though this feature is not yet supported, we already map identifiers to internal numbers denoting VIDs at every host and program (and use them as local database keys).

Attributes are type-value pairs that are associated with a VID, where the type specifies the encoding and the semantics of the value (see section 2.3). For better performance, attributes can be stored for a caching period that is defined per type.

Users can access Virtual ID only through *programs* (see figure 2.1). The program that creates a VID is automatically accredited at this VID with full authorization. Accredited programs can manage your VID and assume your VID towards others. In our simplified setting, their authorization can either be none (which is the default after accreditation), read or write. A program can change the authorization of all less or equally powerful programs (in terms of their authorization) that are accredited at the same VID. Programs authenticate themselves by proving knowledge of a randomly chosen secret and therefore no passwords are needed. Virtual ID does not specify, however, how users are authenticated at a program. Consequently, if a program is accessed over the Web, a password or some other authentication mechanism is still needed there (which is the case for the VID and the Survey Manager).

2.2 Services

The capabilities of Virtual ID can be extended by separate services without requiring an additional concept: The host of a service can be referenced in a corresponding attribute and act itself on your behalf by also being accredited as a program at your VID.

The server is implemented in such a way that handlers can be registered for specific request types. Since services are developed independently from the core, they can be deployed by placing them as JARs (Java Archives) into the folder "~/.VirtualID/Services/", where they are loaded dynamically during runtime.

2.3 Format

For the exchange of information, Virtual ID introduces the *Extensible Data Format* (abbreviated as XDF) that is based on types. Users can specify their own types on the byte-level. Modeling types as VIDs ensures a unique identifier and an optimal integration into Virtual ID.

Conceptually, there are two kinds of types in XDF:

- *Syntactic types* specify the encoding of data and take a fixed or arbitrary amount of semantic types as parameters. Depending on the number of parameters and their purpose, they can be classified as being either final (like string and integer), structural (like list and tuple) or transformational (like compression and encryption).

- *Semantic types* specify the meaning of syntactic types that are instantiated with zero or more semantic types as generic parameters. They can be used to define attributes that have a certain meaning, a certain format and a unique identifier. As an example, a person's name can be modelled in most Western cultures as a tuple of a given name and a family name, where these in turn are based on strings.

As discussed in the conclusion, it was not clear from the beginning how to write a good XDF library. Our current implementation consists of a class that models a block of bytes and a class for each syntactic type that wraps a block for decoding and encoding. Most of the cryptography is done in the wrappers of the encryption and signature types, which results in a modular design.

2.4 Cryptography

All sensitive information is encrypted and signed before it is sent across the Internet. Since requests for information or actions are only sent to hosts, clients can encrypt a random symmetric key with the public key of the host, which signs the response with the corresponding private key and encrypts it with the same symmetric key. The algorithms used in this part of the protocol are AES [6] and RSA [7] respectively.

Programs sign requests differently depending on whether they contact the VID at which they are accredited or another one. In the first case, they provide a non-interactive proof of knowledge of a commitment to a secret value which they submitted during accreditation. In the second case, they attach a non-interactive proof of knowledge of a credential that they obtained from the host of the VID at which they are accredited. Such credentials can be shown anonymously and are described in the paper of Jan Camenisch and Anna Lysyanskaya [2].

Chapter 3

VID Manager

3.1 Description

In order to create and manage VIDs, we provide a program called the VID Manager. Instead of writing a native application for a certain operating system, we decided to offer a simple Web interface that can be accessed from any browser. Virtual ID treats the Web server as a program and therefore does not specify how the user is authenticated at the Web server. To simplify matters, we have chosen the common solution of having a login with a password. Since the user has to choose another password at the Survey Manager, a smarter solution would have been to rely on OpenID to authenticate the user. This is a good example of how Virtual ID and OpenID can complement each other when the former is accessed over the Web.

The VID Manager is available on www.vidmanager.ethz.ch.

3.2 Features

This section names the different features which are realized in the VID Manager.

Create a VID
The VID Manager allows to create VIDs for natural persons at any desired host. For this purpose, the user chooses a unique identifier and the VID Manager attempts to create such a VID at the corresponding host. In order to authenticate users over the Web, they also have to pick a password which is only used at the VID Manager. Besides the identifier and the password, users also choose a name for their VID, which will be stored as an attribute at the newly created VID.

Login with an existing VID
After having created an account, users can login at the VID Manager to change the name of their VID and to manage the accredited programs. These settings are only accessible when the user is logged in, which is checked using cookies.

Change the name attribute
Users can change the name of their VID at any time in the appropriate field. Please note that attributes can be cached and that it therefore can take a while until a change becomes visible to others.

See the accredited programs
After logging in, users can see a list of all programs that are accredited at their VID and have at most the same authorization as the VID Manager. Newly accredited programs enter this list with no authorization. The user can subsequently give it read or write access with the following command.

Authorize a program
As mentioned above, newly accredited programs have no rights initially and therefore their authorization is shown as "None". If you want to allow a program to make requests on your behalf, you need to increase its authorization to "Read". Please note that, for now, all attributes are publicly available, since access control is not yet supported. If you want to allow a program to change the attributes of your VID, then set its authorization to "Write". A program can decrease its own authorization but cannot give another program a higher authorization than its own.

Remove a program
Whenever there are accredited programs that you no longer want to use, you can remove them as long as their authorization is lower than or equal to the one of the VID Manager. Programs without an authorization cannot remove other programs because they cannot even access the list of accredited programs.

Chapter 4

Survey Service

4.1 Description

This project is the reference implementation of the Survey Service, which is an extension to Virtual ID that provides anonymous online surveys. As described in section 2.2, the server is designed in such a way that the handlers for the newly specified request types can easily be registered during initialization.

In order to exchange surveys and their submissions in the Extensible Data Format, we specified many new semantic types. Being a protocol that is built on top of Virtual ID, the Survey Service reuses its communication and authentication mechanisms (which is no necessity for services).

4.2 Cryptography

In order to submit a survey anonymously but only once, the hosts of the participants requests a credential from the issuer of the survey that contains a random token. This token is hidden from the issuer during the issuance of the credential but exposed on the submission of the survey. Consequently, the issuer can detect when a participant submits the survey twice. Since the same credential mechanism is used as in the core protocol of Virtual ID, it is also possible to combine the two (as described in the outlook). Please note that the Survey Service is not an e-voting protocol and that participants cannot verify the evaluation of a survey.

Chapter 5

Survey Manager

5.1 Description

The Survey Manager is a program that allows you to manage your surveys of the Survey Service as described in chapter 4. It works in the same way as the VID Manager, i.e. it is also accessed over the Web and the user also needs to choose a password for authentication.

As a user, you can either create a survey, which is pushed to the participants, or participate in a survey that you received. We call the former ones outgoing surveys and the latter ones incoming surveys. All features of the Survey Manager are documented in the following section.

The only question type that is currently supported is a rating with five options. This is typically the kind of question used in the evaluation of courses at universities. The available grades are '- -' (very bad), '-' (bad), '=' (satisfiable), '+' (good), '++' (very good) and 'nA' (not answered). At the end of every survey is a field for arbitrary comments.

The Survey Manager is available on www.surveymanager.ethz.ch.

5.2 Features

This section presents the various features of the Survey Manager.

Create a Survey Manager account
Before being able to use the Survey Manager, one has to create an account at the Web server constituting the Survey Manager. For this purpose, users provide the identifier of an existing VID and a password in the corresponding fields.

Login with an existing Survey Manager account
After having created a Survey Manager account, users can login to manage their surveys with the following features.

List the incoming surveys
After logging in, users can see all the surveys to which they were invited and which they have not yet submitted.

List the outgoing surveys
This feature lists all the surveys that the user created.

Fill out and submit an incoming survey
Incoming surveys can be filled out and submitted to the issuer. Every question can be rated by chosing one of the available options. Furthermore, general comments can be added at the end of an incoming survey. Since a survey can only be submitted once, it is no longer shown after its submission.

Create an outgoing survey
Users can create surveys with an arbitrary amount of questions, which they can send afterwards to a fixed set of participants (i.e. participants cannot be added or removed later on). Participants are entered with the identifier of their VID. For pushing a survey to another VID, the own host needs to look up the attribute that references the host which is responsible for the surveys of the other VID.

See the evaluation of an outgoing survey
The issuer of an outgoing survey can see its current evaluation at any time. Since surveys are pushed to all participants simultaneously and submissions are anonymous, the issuer of a survey cannot know which participant submitted which answers.

Chapter 6

Conclusion

6.1 Results

Even though we left out many concepts and features of Virtual ID, we have made a proof of concept that can be used as a starting point for a real reference implementation. The following concepts, which are described in the paper by Kaspar Etter [1], are not supported by our current implementation: Certificates, contacts and contexts, and roles. As partially mentioned, we also skipped the relocation of VIDs, the grouping of attributes and the fine-grained authorization of programs, since all attributes are publicly accessible anyway. Finally, we also do not achieve the liability of hosts that is desired in the ultimate protocol.

6.2 Problems

During the implementation of the projects documented in this report, we faced the following two problems:

Bootstrapping
Several times during the implementation phase, we came across bootstrapping issues. Thanks to the many assertions that we had inserted into our projects, these problems could easily be detected but finding appropriate solutions was more challenging. For example, we had written the configuration files in our Extensible Data Format (XDF) but, since XDF itself is based on Virtual ID, we would have already needed a running server when starting one. We fixed this issue by dropping the type information from the data blocks and by removing all associated checks, so that we were able to decode and encode XDF types without performing any lookups.

XDF implementation
We have spent quite some time on the implementation of a nice XDF library with a reasonable performance. The challenge consisted in decoding and especially encoding XDF types without copying byte arrays unnecessarily. We achieve this by encoding types lazily, i.e. we wait with the actual encoding for as long as possible in order to write the block directly into its final memory location.

6.3 Outlook

Being just an example service, we kept the Survey Service as simple as possible. We could, however, imagine some nice cryptographic enhancements like an onion routing on the host level to anonymize submissions also on the network level, providing certified attributes like age, gender or nationality anonymously during submission or compensating participants by drawing lots without violating their privacy, which could be accomplished with verifiable encryption.

Last but not least, implementing Virtual ID for the first time has given us valuable insight into which aspects of the protocol work well and which ones need to be improved.

Bibliography

[1] Etter, K.: Virtual id, bachelor thesis at eth zurich. http://www.virtualid.ethz.ch (2012)

[2] Camenisch, Lysyanskaya: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: EUROCRYPT. (2001)

[3] : Openid authentication 2.0. http://openid.net/specs/openid-authentication-2_0.html (2007)

[4] : Openid attribute exchange 1.0. http://openid.net/specs/openid-attribute-exchange-1_0.html (2007)

[5] : The oauth 2.0 authorization framework. http://tools.ietf.org/html/draft-ietf-oauth-v2-26 (2012)

[6] : Announcing the advanced encryption standard (aes). http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf (2001)

[7] Rivest, S., Adleman: A method for obtaining digital signatures and public-key cryptosystems. http://people.csail.mit.edu/rivest/Rsapaper.pdf (1978)