Castella Design Rationale

Justifying Design Choices for Card Holder Authentication Caching

29 nov 2005

Revision History
Revision 1.1 29 nov 2005
Converted to docbook
Revision 1.0 15 nov 2005
Initial Release

Table of Contents

1. Introduction

Smart cards[1] can be used in an ever increasing number of applications. This creates convenience and security issues. Convenience issues, because having every application ask for the PIN code can become a burden. Security issues, because risk of PIN exposure increases with the number of times it is entered.

Castella is a solution to these problems. Instead of entering the PIN in every application, PIN entry is performed only once. This is referred to in the rest of this document as Card Holder Authentication Caching[2]. A user can continue to use the card until it is removed from the smart card reader, or until the PC is not used for a certain period of time. In this last case authentication to the card will be dropped automatically.

In this document both the reasons for deploying Card Holder authentication caching and the design of Castella which implements it will be described.

The document will first give a definition of what Card Holder authentication exactly is (Section 2, “Definitions” [1]). Secondly, the caching aspect will be investigated, defining the concept of caching and addressing the reasons for caching (Section 3, “Reasons for Caching” [1]). Subsequently (Section 4, “Design” [1]), the design of Castella will be detailed, including such topics as the different components and how they relate, the security mechanisms that are part of Castella and finally, the value of the design and the effects of Card Holder authentication caching in general.

2. Definitions

Card Holder authentication is the act of authenticating to the card, e.g. entering the PIN code for a Smart Card. A correct PIN code allows the card holder to use the card for cryptographic operations.

Caching card holder authentication is the act of keeping around the security context previously established by a user successfully authenticating to the card. Consequently, when a user wants to perform another cryptographic operation with the smart card, this is allowed because of the cached security context. Caching will be in effect accross different applications, independent of whether the applications are running locally or within a remote access session on a remote system.

Exact details about the design of Card Holder Authentication Caching can be found in the section about Component Design (Section 4.3, “Component Design” [2]).

3. Reasons for Caching

There are four reasons for deploying authentication caching:

1. It adds convenience, especially when the Smart Card is used by multiple applications.
2. It inhibits the transmission of PINs accross the network during remote system access. See Component Design section (Section 4.3, “Component Design” [2]) for details.
3. It reduces risk of PIN exposure to onlookers by decreasing the number of times card holder authentication is performed.
4. It allows implementation of centralised Smart Card related policies that are enforced for all applications. Caching in itself is just an aspect of such a policy.

4. Design

This section details the design of Castella, investigating the following subjects:

- The purpose of Castella (Section 4.1, “Purpose” [2])

[1] Note that the functionality that Castella provides is not limited to (smart) cards, only, but also applies to e.g. USB tokens.
[2] Card Holder authentication caching is also known as PIN caching. However, because Castella also supports authentication with a fingerprint, PIN caching is not an accurate description. Furthermore, PIN caching suggests that a PIN is kept in memory somewhere and that is especially not the case with Castella.

- The requirements Castella must adhere to. (Section 4.2, “Requirements” [2])
- A detailed overview of the components and the relations between the components, paying partcular attention to the security mechanisms in the design (Section 4.3, “Component Design” [2])
- The value of the design (Section 4.4, “Value of the Design” [3]).

4.1. Purpose

Increasing convenience and security by coercing applications into deferring Card Holder Authentication to a centralized service. This centralized service implements a caching policy allowing applications to share Card Holder authentication.

4.2. Requirements

The following list of requirements is a design oriented rehash of the reasons for caching from (Section 3, “Reasons for Caching” [1]). The design is required to:

1. Reduce the number of times Card Holder authentication is performed.
2. Inhibit a PIN code from travelling accross the network.
3. Drop authentication when the Card Holder isn't using his PC.
4. Use a centralized policy to implement caching of Card Holder Authentication to share it between different applications.
5. Prevent other users from stealing a cached Card Holder Authentication.
6. Not keep a PIN code or any other sensitive authentication information in memory that can be used to perform Card Holder Authentication.

4.3. Component Design

Castella is comprised of two components, a library (is something that can be installed into an application to extend its behavior) and a service (is a program that runs in the background and performs operations on behalf of the user).

The Castella library implements several cryptographic interfaces. This allows it to be installed as an add-in or as a security module in different cryptographic applications. The SafeSign Cryptographic Service Provider (SafeSign CryptoAPI CSP) for example will load the Castella library, forcing all applications created by Microsoft to use Castella.

The Castella service needs to run on a card holder PC only. So although it may also be installed on a remote access server, this is only required when a card holder will sign-on to that system locally using the smart card.

The Castella service uses the SafeSign cryptographic token interface library (SafeSign PKCS#11) to access the Smart Card.

4.3.1. Castella Library Design

The Castella Library implements several interfaces, making it a multi-functional library that performs the following functions:

1. It's an add-in for SafeSign CryptoAPI CSP, forcing applications like Internet Explorer, Outlook (Express) and Microsoft VPN to use Castella.
2. The library is a security module for PKCS #11 enabled applications like FireFox, Mozilla, Netscape and Entrust. It gets installed instead of the SafeSign PKCS#11.
3. It's an add-in for Citrix MetaFrame Client to connect to a MetaFrame server and allow for authentication forwarding to remote applications. The Castella library also creates a bridge allowing a Castella library on a remote system to connect to the Castella service through this add-in.
4. It's an add-in for Windows Terminal Server Client to connect to a Windows Terminal Server (WTS) or to Windows XP professional. The Castella library also creates a bridge between the remote Castella library and the local Castella service.
5. It's a WinLogon notification package to detect login & screensaver activation. The events will be forwarded to the Castella Service to allow it to make policy decisions. For example, the screensaver activation is used to drop Card Holder authentication on the smart card.
6. It's an add-in for the SafeSign CryptoAPI store provider which keeps track of the available user certificates. It caches certificates (public information) for the store provider to reduce the load on the Card Holder's Smart Card in situations where multiple remote access session are trying to access it.

The Castella library connects to the Castella Service in one of three ways:

1. Using a Windows NT Named Pipe. A native operating system service that allows applications to

communicate. Named Pipes can be secured. This allows the Castella Service to determine whether a user connecting is the same one that originally performed Card Holder Authentication.

2. Using a Windows Terminal Services Virtual Channel. This is a native Terminal Services operating system service that can be used to communicate between applications running in a remote session with applications running on the Client PC that initiated the connection.

3. Using a Citrix MetaFrame Virtual Channel. This is a service available in a Citrix MetaFrame server that has the same properties as Windows Terminal Services Virtual Channels.

The Castella library is smart in that it can determine whether it is running in a remote session or whether it is running locally. The library connects to the Castella service using a named pipe when it is not running in a remote session. Consequently, when it is loaded in a remote session, the library will automatically use the correct type of virtual channel (Citrix or WTS) to connect to the Castella Service.

In the case that a connecting remote access client does not have the Castella library add-in installed, a virtual channel cannot be opened to create a bridge to the Castella Service. In this case the Castella Library will be come transparent and behave as the normal SafeSign PKCS#11 library by forwarding all calls directly to library installed on the terminal server.

To help in making policy decisions in the Castella Service, the library will send the name of application that loaded it to the Castella service. When the Castella library is working as a bridge by being loaded in both the remote access client as well as in an application on a remote system, the service will receive both application names. So it can determine which remote access client was used and which application on the remote system is using the Castella library.

4.3.2. Castella Service Design

The first time a user performs Card Holder authentication an authenticated PKCS #11 session to the smart card will be kept open. The SafeSign PKCS #11 library will keep an authenticated logical channel open to the smart card for this session. When the same user subsequently uses an application, it will reuse the already open session. So there is no need for a PIN code to be stored by either Castella or SafeSign.

The Castella service is accessable through securable named pipes. Whenever a Card Holder successfully authenticates, a Dynamic Access Control List (DACL) is created that allows only that Card Holder to access the named pipe. Whenever a different user establishes a new named pipe connection, the Card Holder authentication is dropped, but the connection is allowed. When the first user becomes active again, the authentication is again dropped. This implementation prevents sharing of Card Holder Authentication by different users, but allows things like user switching. The crux of this design is that the Windows operating system knows which user is associated with initiating a named pipe connection and that the service is capable of acquiring that information.

For connections coming from remote access clients (Citrix MetaFrame and Windows Terminal Server) the Castella service will report that the Card Holder authentication is done by protected authentication path. A protected authentication path is a device that is connected directly to the smart card to perform Card Holder Authentication. The unique aspect being that the authentication information never leaves the device. So setting the protected authentication path flag will force applications running in remote sessions never to actually ask for a PIN code and sent it accross the network.

When the service receives a notification that the screensaver has activated, the Card Holder authentication that is valid at that moment will be dropped.

4.4. Value of the Design

The value of the design can be summarised as follows:

- Works with protected authentication path devices like PIN pad readers.
- Reduces number of Card Holder authentications significantly.
- Forces remote applications to not ask for and send a PIN code accross the network.
- Improves speed when multiple remote sessions are trying to access the card by caching certificate registration information.
- Placing the policies in the Castella Service allows for enforcement of policy rules in a central place.

5. Conclusion

The use of Castella improves usability, while making no concessions to security.

Keeping the PIN entry confined to interaction between Castella and the smart card will reduce the risk of exposure associated with the PIN traveling accross the network.

Decreasing the number of times the PIN is entered will reduce the risk of exposure to onlookers.