Authentication Framework for Smart Cards

Apostol Vassilev, Michael Hutchinson

Smart Card R&D
Schlumberger
8311 North FM 620
Austin, TX 78726, USA
vassilev@slb.com
hutchinson3@slb.com

Abstract: This paper introduces a generalized authentication framework for smart cards. The framework abstracts the authentication services on the card and allows flexible configuration of authentication policies and technologies. This paper also makes recommendations for extensions to current authentication APIs.

1. Introduction

User authentication is one of the critical elements guaranteeing the security and integrity of public key infrastructure (PKI) solutions based on smart cards. Recent progress in smart card technology, in particular the rapid gain in popularity of Java Card technology [SunJC1, SunJC2, SunJC3, SunJC4] among providers and customers of IT security solutions, has created new opportunities and challenges for implementation of flexible and secure user authentication in the context of the Java Card Virtual Machine (JCVM) [SunJC2].

The classic approach to smart card user authentication is based on a Personal Identification Number (PIN). The JCVM allows any applet instantiated on the card to implement its own user authentication scheme to protect its data from host applications or from other applets on the card. The typical implementation is based on the PIN class from the javacard.framework package, and usually each applet contains an instance of a PIN object that is used to verify and maintain user credentials.

As the feature set of JCVM is extended beyond Java Card version 2.1 and the hardware capabilities of smart cards are enhanced further with much bigger memory resources and more powerful processors, the complexity and capability of the Java Card Runtime Environment (JCRE) [SunJC3] increases accordingly. As a result, smart card user authentication must rise to a new level, allowing flexibility and support of sophisticated security policies required by organizations from the military, law enforcement, financial, and general business spheres.

This paper is organized as follows. In Section 2, we introduce the background issues that explain why we require the framework. In Section 3, we introduce new authentication framework architecture for smart cards. In Sections 4, 5, and 6, we discuss the roles and requirements for the essential elements of the new architecture. In Section 7, we discuss the framework security and integrity. Section 8 contains conclusions.

2. Background and Issues

In the context of PKI applications, the smart card is used as a security token that provides a higher degree of security for applications running on the host computing platform. The typical scenario is that users need to authenticate to the computer system using a smart card before they are allowed to use resources on the machine or network. Host applications interact with applications (applets) on the smart card in order to establish the user's identity and make use of the user's credentials after authentication is established.

A growing number of PKI application scenarios demand a level of flexibility that emerging user authentication solutions must address. For example:

- Several different applets on the card might execute concurrently in the JCRE. Each applet might implement separate user authentication, or applets might share authentication; that is, if the user is authenticated to one applet in a defined set of applets, the user is authenticated to all applets in that set.
- Organizations might wish to configure smart cards for use by more than one user, for example, to give access to a card to the site security officer to perform card maintenance operations, or to configure cards for secret sharing when operations (e.g., large financial transactions and certain military actions) require more than one person's authorization.
- An important trend in user authentication is the demand for alternative authentication technologies, moving beyond a simple PIN. An applet on the card might, for example, require user authentication by a fingerprint, or by a combination of fingerprint and PIN. A variety of biometric technologies have emerged in the marketplace, some of which are ideally suited for implementation on smart cards (Match on Card, or MoC) [JCBio22, Pet01], while some are more suited for implementation on host machines.
- Biometric technologies vary in terms of the human biological features they measure for authentication (DNA, iris, retina, fingerprint, voice, face, dynamic of keyboard typing, etc.), as well as the amount and format of data communicated between the biometric reading device and the device or system performing the authentication. Biometrics standards, like the "Biometric API" [Bio01] (promoted by the BioAPI consortium) or standards promoted by CBEFF [NIST01] and others, do not identify specific technologies or specify how to use multiple technologies simultaneously on the card.

The authentication security policy has, until now, been designed and implemented in each applet on the card. This makes it difficult, if not impossible, to update existing applets or incorporate applets from third party vendors, when these new authentication technologies emerge.

This limitation has been a major problem for system integrators who often want to deploy new applets without disturbing existing ones. There is a need for a general authentication framework design on the smart card that abstracts the authentication service from the functional applets and allows flexible configuration of authentication policies and technologies.

An authentication framework will improve the security of smart cards, simplify the maintenance of functional applets with respect to reconfiguring the authentication policy, and allow the addition of new authentication technologies on the card.

When more than one authentication technology is present on the card, it is desirable to identify both the type of technology provided by an applet and the individual the technology identifies, yet current standards do not define how to obtain such information.

It is a well-known fact from the field of biometric authentication that the match of a candidate and reference template is never exact. Contributing factors are temporary or permanent deviations in the biometric characteristics of the person, changes in the sensitivity of the measuring equipment, etc. Currently, the Biometric API [Bio01] implements a threshold level for the match which generates a Boolean Pass/Fail result from the authentication. This threshold is associated with one instance of the applet performing the match. If more than one level is required for access to different secure objects, multiple instances of the biometric authentication applet are needed, each with its own copy of the user template data. Clearly, this is not a very efficient use of the limited resources of the card.

3. Authentication Framework Architecture

For the purposes of the discussion in this section, we consider three types of card applications:

- **Card Application Applet (CAA)** – an applet on the smart card needing authentication for accessing protected data within the applet. Typically, the CAA defines roles (e.g., user or security officer) and assigns certain privileges to them (e.g., a user can access the exchange private key, or a security officer can unblock the user PIN). Examples of CAAs are a bank wallet applet or an applet providing services required by a smart card-based PKI system (e.g., certificate storage, public key encryption, or signature).

- **Authentication Technology Applet (ATA)** – an applet providing specific technology for holding and verifying the user's identity. Examples of ATAs are applets providing PIN verification or fingerprint match.

- **Authentication Policy Applet (APA)** – an applet responsible for providing user authentication and authentication security policy services to CAAs on the smart card as well as to host applications. This applet bridges a CAA with one or more ATAs in order to provide authentication services for the roles CAAs implement according to the security policy established on the card. This bridge links a role defined by a CAA to a user identity contained by one or more ATAs. The APA is the core element of the authentication framework on the smart card.

One important factor for defining the architecture of the authentication framework is that for the purposes of a CAA, the problem of authenticating the user can be reduced to the basic question "**Is the user authenticated?"** If **yes**, then grant access to protected data. If **no**, deny access. The details of what represents the user's identity (PIN, fingerprint, etc.) or how it is checked and maintained are issues that can and should be separated from the main functional specification of the CAA. Delegating the tasks of authenticating the user and maintaining an authentication policy with the ATA simplifies the CAA design, implementation, and maintenance. Defining clean CAA/APA and APA/ATA interfaces provides the possibility to reconfigure and adopt new authentication technologies without any changes to the CAA.

Note. *In the following discussion, we assume the security model of the JCRE, which utilizes firewalls and sharable interfaces to protect access to applet data and methods. However, the general idea of the three-tier framework architecture applies to other smart card computing platforms, such as .NET [NET02, NET03].*

The architecture of the authentication framework for Java card smart cards is shown in Figure 3.1.

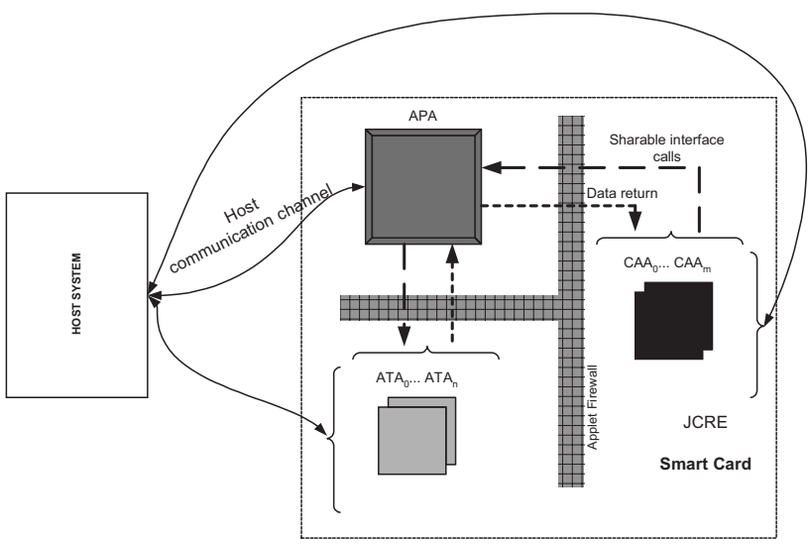

Figure 3.1. Authentication Framework Architecture

To simplify the presentation, we consider only one card application applet (CAA) and one technology applet. The key element in this architecture is that we have abstracted the user authentication from the CAA into a separate applet (ATA) that will be responsible for checking and updating the user identification record (e.g., a PIN or biometric data) based on the specific technology implemented by the applet.

The design in Figure 3.1 shows that the CAA code does not have to change at all in order to replace a PIN-based ATA with a fingerprint-matching ATA, assuming that it provides the sharable applet interfaces required by the framework. The code of the APA also does not need to change; only the data it contains must be updated to reflect the switch to the new ATA. Neither the CAA nor the APA need to know anything about the specifics of data exchange between the ATA and the host application that typically allows the user identification record (a PIN or a biometric template) to be sent to the ATA for verification.

The design shown in Figure 3.1 emphasizes that information exchange among the main architecture components (APA, CAA, and ATA) occurs through sharable applet interfaces only. The data objects of any CAA are separated from those of the APA or any ATA by the JCRE firewall. However, the data objects of a particular CAA might or might not be protected from other CAAs with a JCRE firewall. Normally, this is a design decision made by CAA developers.

4. Authentication Policy Applet (APA)

The Authentication Policy Applet (APA) is the central component of the authentication framework for Java cards (cf., Figure 3.1). It contains the high-level authentication policy setup and provides the bridge between a CAA and its set of ATAs.

The policy setup might be organized as follows:

A CAA designer might incorporate several different security requirements into the CAA. Each security requirement might be associated with a particular purpose, e.g., “exchange key protection”, “signature key protection”, or “large transaction shared secret”. Naturally, security requirements vary from one CAA to another. Consequently, “exchange key protection” of CAA_i might be different from that of CAA_j. On the other hand, if the applet designers want to use the same security configuration, it should be possible to configure both CAA_i and CAA_j to do so.

The APA will maintain a list of security configurations. Each of these holds a Boolean expression detailing which identities need to be authenticated to allow access to data it protects.

For example, a “large transaction shared secret” might require a security configuration as follows:

((card holder authenticated by a *PIN*) **AND**
  ((department manager authenticated by a *PIN*) **AND**
    (department manager authenticated by a *fingerprint*))).

Alternatively, the last clause can be modified to use the match score as:

((department manager authenticated by a *PIN*) **AND**
  (department manager authenticated by a *fingerprint with a match exceeding 80%*)).

Note that in this case, the ATA needs to report the score. The APA checks the score against the threshold value established by the high-level policy. The utilization of the match score allows the security policy to handle complex configurations such as:

((card holder authenticated by a *fingerprint with a match exceeding 80%*) **OR**
  ((card holder authenticated by a *PIN*) **AND**
    (card holder authenticated by a *fingerprint with a match exceeding 50%*))).

In the cases of hardware failures, diverse user population, or other deployment problems, a policy like this can greatly improve the usability and robustness of the smart card PKI infrastructure while maintaining a high security threshold.

5. Authentication Technology Applet (ATA)

An Authentication Technology Applet (ATA) provides specific Match on Card (MoC) technology for holding and verifying the user's identity. Examples of ATAs are applets providing PIN verification or fingerprint match. In this section, we consider some of the functional behaviors and integration requirements for such applets.

Since an ATA interfaces with host applications, it must ensure proper operation in the context of a multi-process environment on the host. In cases like this, the ATA must allow a host process to identify itself by associating a secret (an ID) with a successful authentication. The ATA must maintain a separate authentication state for each ID. This ID must be a cryptographically strong number, different from the integer process ID typically assigned to processes by the host computer's operating system because process IDs are easy to impersonate. The ID needs to be negotiated and exchanged using some cryptographic algorithm for secret exchange [MOV97] between the host application and any of the CAAs and/or ATAs that the host is engaging. It is the responsibility of the processes on the host to share or protect the ID established by a given process for communicating with a CAA and/or an ATA from the framework.

Successful validation of user identification record (PIN, fingerprint, etc.) sets the validation state to true. For validations that involve biometrics, validation might mean that the user sample exceeds a configurable threshold value. Low-level, ATA-specific security policies may control the properties of the identification record (e.g., number of characters in the PIN, least number of specific type characters, or which finger to use) or the transition of the verification state. For example, some policies might require that the validation state be reset after the first query.

The host process must supply an ID when it authenticates to the ATA. When the CAA queries whether the ID has been authenticated, the ATA will be able to match the ID supplied by the CAA with the IDs authenticated in the ATA, and respond accordingly.

In view of the potential use of different MoC technologies, the ATA must offer a mechanism to provide the type of technology used.

To improve the efficiency of the framework in the context of variable match score implementations, the ATA must have a mechanism to provide a percentage score associated with the match achieved. This allows multiple threshold levels to be defined by a high-level APA security policy rather than the ATA, which eliminates the need for multiple instances of the ATA.

To accommodate the need for multi-user smart card applications, the ATA must have a mechanism to provide the name of the user whose identification record the ATA contains.

6. Card Application Applet (CAA)

The CAA is an applet on the smart card that requires authentication to access protected data within the applet. In this section, we discuss the practical implications of integrating existing CAAs into the framework.

Only two important changes are needed to accommodate the framework design. First, all internal (to the applet) authentication code must be replaced by calls to the APA for the authentication state. Second, if the host operating environment requires differentiating processes on the card, the CAA must change to accept a secret (an ID) from the host and submit it to the APA in queries of the authentication state. High-security applications may require mutual authentication (cf., [MOV97]) between the host and the CAA prior to the secret exchange to protect it from being disclosed to unwanted parties.

7. Framework Security and Integrity

The security and integrity of the authentication framework is dependant upon the card security system for installation to the card, configuration of the policies, execution, and maintenance. For maximum protection, the APA, ATAs, and CAAs should be added to the card in a secure environment. The deploying organizations must follow established security processes for storing and maintaining keys that allow access to protected resources on the card. This, along with the security of the card operating environment, will ensure the integrity of the framework and prevent component impersonation attacks.

Once configured, the framework should use secure channel communication with the host to prevent adversaries from being able to steal sensitive information flowing between the host and card. Clearly, the security of this interaction also depends on the security model for processes within the host operating system.

8. Conclusions

The proposed framework abstracts the authentication services on the card and allows flexible configuration of security policies. The three-tier framework architecture allows modification of existing or addition of new authentication technologies on the card.

System integrators who want to take advantage of new technologies without disturbing existing ones can easily achieve this through the framework.

Developers can use the framework architecture to improve the security, efficiency, and maintainability of their software. The framework security model fits well into the security model of the card operating system. It imposes only standard and well-understood requirements for developers to ensure the integrity of the framework and prevent attacks against it.

The framework imposes minimal requirements for extending existing authentication technologies standards to improve their interoperability and acceptance.

Bibliography

[Bio01] Bio API Consortium: BioAPI Specification Version 1.1. 2001. www.bioapi.org.

[JCBio22] Java Card Forum: Java Card 2.2 Biometry API proposal www.javacardforum.org/Documents/JCFBioAPIV1A.pdf

[MOV97] A. Menezes, P. van Oorshcot, S. Vanstone: Handbook of Applied Cryptography. CRC Press, New York, 1997. ISBN 0-8493-8523-7.

[NET02] Schlumberger: .NET Card Technology. 2002. www.smartcards.net/infosec/.NET.html

[NET03] Hive Minded Inc.: Smartcard.NET – Technology overview, Version 1.0. 2003. www.hiveminded.com/whitepapers/.

[NIST01] The National Institute of Standards and Technology: Common Biometric Exchange File Format (CBEFF), NISTIR 6529. 2001. www.itl.nist.gov/div895/isis/bc/cbeff/

[Pet01] M. Pettersson: Match On Card Technology. Precise Biometrics White Paper. Precise Biometrics, Inc. 2001. www.precisebiometrics.com

[SunJC1] Sun Microsystems, Inc.: Java Card 2.1 Platform Specifications. http://java.sun.com/products/javacard/specs.html

[SunJC2] Sun Microsystems, Inc.: Java Card 2.1 Virtual Machine Specification. http://java.sun.com/products/javacard/specs.html

[SunJC3] Sun Microsystems, Inc.: Java Card 2.1 Runtime Environment Specification. http://java.sun.com/products/javacard/specs.html

[SunJC4] Sun Microsystems, Inc.: Java Card 2.1 Application Programming Interface Specification. http://java.sun.com/products/javacard/specs.html