



OATH Reference Architecture, Release 2.0 **Initiative for Open AuTHentication (OATH)**

The Initiative for Open AuTHentication (OATH) welcomes input, suggestions, and other feedback on this work from as broad a range of industry participants as possible, in order to improve its quality. Feedback should be sent to oath_technical@v2.listbox.com.

If you are interested in getting more information about OATH or joining OATH, please contact info@openauthentication.org or visit <http://www.openauthentication.org>.

Copyright(c) 2004-2007, Initiative for Open AuTHentication. All Rights Reserved.

CONTENTS

1. Executive Summary	5
2. Abbreviations	6
3. OATH Vision and Goals	8
4. Usage Scenarios	10
4.1. Remote Access	10
4.2. Online Banking	10
4.3. Telecommuting	10
4.4. Client and Business Partner Extranet	11
4.5. eGovernment	11
4.6. 24x7 IT Infrastructure Support	11
4.7. Wireless Roaming	11
4.8. Desktop Logon	11
4.9. Closed network	12
5. Authentication Framework	13
5.1. Client Framework	13
5.2. Provisioning and Management Framework	13
5.3. Validation Framework	14
5.4. Applications	14
5.5. Authorization	14
5.6. User Store	14
5.7. Policy Store	14
5.8. Audit Store	14
5.9. Authentication and Identity Sharing	14
5.10. Risk evaluation and sharing	15
6. OATH Reference Architecture	16
6.1. Client Framework	16
6.1.1. High-Level Architecture	16
6.1.2. Salient Features	17
6.1.3. Authentication Methods	18
6.1.4. Authentication Tokens	19
6.1.5. Token Interface	20
6.1.6. Authentication Protocols	21
6.2. Validation Framework	22
6.2.1. High-Level Architecture	22
6.2.2. Salient Features	24
6.2.3. Existing Standards and Technologies	25
6.2.4. OATH Focus Areas	26
6.3. Risk evaluation and sharing framework	26
6.3.1. High-level architecture	26
6.3.2. Salient features	27
6.3.3. Existing Standards and Technologies	28
6.3.4. OATH Focus Areas	28
6.4. Provisioning and Management Framework	28
6.4.1. High-Level Architecture	29

6.4.2.	Salient Features	30
6.4.3.	Existing Standards and Technologies	31
6.4.4.	OATH Focus Areas.....	32
6.5.	Common Data Model.....	33
6.5.1.	Existing Standards and Technologies	33
6.5.2.	OATH Focus Areas.....	34
6.6.	Authentication and Identity Sharing.....	34
6.6.1.	Authentication Sharing	35
6.6.2.	Identity Sharing.....	38
6.6.3.	Traditional Federated Identity.....	38
6.6.4.	User-centric Identity Sharing.....	39
6.6.5.	OATH Focus Areas.....	40
7.	Example Deployment Scenario.....	41
8.	Summary of OATH Focus Areas.....	44
9.	References.....	46
10.	Contributing members	49

1. Executive Summary

This document specifies version 2.0 of the reference architecture for the Initiative for Open AuTHentication (OATH). The OATH Reference Architecture document describes a high-level technical framework for open authentication, as envisioned by the OATH member companies.

The reference architecture is intended to explain OATH's vision for authentication, as well as to provide a high-level technical roadmap for its work. The intended audience includes decision makers and technical architects from OATH member and nonmember companies, IT managers and architects from organizations that are considering deploying strong authentication solutions, and other standards organizations that share all, or part, of the OATH vision.

The work has been driven by the following guiding principles:

- **Open and royalty-free specification** - OATH intends to establish an open and royalty-free specification for strong authentication by leveraging existing open standards, where possible, and leading standardization efforts in well-established technical standards bodies where existing standards are not available.
- **Device innovation and embedding** - OATH intends to specify components for low-cost, multi-function authentication devices (e.g. tokens and smart cards that can support multiple authentication methods) and transform today's mobile devices (e.g., mobile phones, PDAs, laptops, etc.) into strong authentication devices.
- **Native platform support** - OATH intends to facilitate native support (e.g. platform connectors) for strong device and user authentication in application and identity management platforms. It also intends to leverage existing infrastructure building blocks, such as LDAP directories and AAA servers.
- **Interoperable modules** - OATH intends to enable best-of-breed solutions through a framework of interoperable components. In this way, a user organization will be able to deploy modules (both software and service-based) and devices (both hardware and embedded) from different vendors, in a comprehensive authentication solution.

An open reference architecture, such as the one described in this document, will serve as a powerful tool for fostering competition and innovation among key solution providers, such as device manufacturers, identity management vendors, security service providers and application developers. It will, consequently, lower the complexity and cost of deploying strong authentication in applications, thereby helping to realize the OATH vision of "universal" strong authentication.

The rest of this document is structured, first of all, to provide a context for the work, in the OATH Vision and Goals Section (Section 3) and the Usage Scenarios section (Section 4). The high-level Authentication Framework is then introduced (Section 5). The next six sections provide greater detail in the areas upon which OATH is planning to focus, viz. Client Framework (Section 6.1), Validation Framework (Section 6.2), Risk evaluation and sharing (Section 6.3), Provisioning and Management Framework (Section 6.4), Common Data Models (Section 6.5) and Identity-Sharing (Section 6.6). The document concludes by describing an example deployment scenario (Section 7) and by summarizing the OATH focus areas (Section 8).

2. Abbreviations

AAA	Authentication, Authorization and Auditing
CA	Certification Authority
CAP	Chip Authentication Program
CMC	Certificate Management Protocols over CMS
CNG	Microsoft Cryptographic API: Next Generation
CMP	Certificate Management Protocols
CMS	Cryptographic Message Syntax
CRL	Certificate Revocation List
CRM	Customer Relationship Management
CRMF	Certificate Request Message Format
CTKIP	Crypto Token Key Initialization Protocols
DLOTA	Download Over-the-Air Protocol
DRM	Digital Rights Management
EAP	Extensible Authentication Protocol
EIS	Enterprise Information Systems
EMV	Europay MasterCard Visa
ERP	Enterprise Rights Management
GSM	Global System for Mobile Communications
GSSAPI	Generic Security Service Application Program Interface
GUI	Graphical User Interface
HTTP	Hyper Text Transfer Protocol
HOTP	HMAC-based OTP
IdP, IDP	Identity Provider
IETF	Internet Engineering Task Force
I/O	Input / Output
IPsec	Internet Protocol Security
ISP	Internet Service Provider
JAAS	Java Authentication and Authorization Service
LDAP	Lightweight Directory Access Protocol
MIDP	Mobile Information Device Profile
MINA	Multipurpose Infrastructure for Network Applications
OATH	Initiative for Open AuTHentication
OCRA	OATH Challenge Response Algorithm
OCSP	Online Certificate Status Protocol
OMA	Open Mobile Alliance
OTA	Over-The-Air
OTP	One-Time-Password
PAM	Pluggable Authentication Module
PAP	Password Authentication Protocol
PDA	Personal Digital Assistant
PKCS	Public Key Cryptography Standards
PKI	Public Key Infrastructure
PSKC	Portable Symmetric Key Container
RADIUS	Remote Authentication Dial In User Service
RFC	Request for Comments
SAML	Security Assertions Markup Language
SASL	Simple Authentication and Security Layer
SCEP	Simple Certificate Enrollment Protocol
SI	System Integrator
SIM	Subscriber Identity Module
SME	Small and Medium Enterprise
SMS	Short Message Service

SPEKE	Simple Password-authenticated Exponential Key Exchange
SSL	Secure Sockets Layer
TACACS	Terminal Access Controller Access Control System
TDS	Transaction Digital Signing
TPM	Trusted Platform Module
TLS	Transport Layer Security
XKISS	XML Key Information Service Specification
XKMS	XML Key Management Specification
XKRSS	XML Key Registration Service Specification
USB	Universal Serial Bus
VoIP	Voice over Internet Protocol
VPN	Virtual Private Network

3. OATH Vision and Goals

More and more, organizations are leveraging the benefits of e-business by opening up their networks and applications for access by employees, business partners and customers. The traditional method of securing that access, the static password, is coming under increased scrutiny by IT and business executives, as they come to the realization that password-based access-control, by itself, cannot meet their information security and regulatory compliance requirements. A stronger digital identity must be issued to information system users, in order to ensure that valuable information is not accessed, modified or stolen by unauthorized users, and to ensure that the security, privacy and reporting requirements of various industry regulations are met.

For this reason, OATH believes that the path toward strong digital identity must start with strong authentication.

Strong authentication is the first pillar of trusted networks, in which identities can be trusted by independent partners. It is the foundation for a more secure network, where all users and all devices are strongly and mutually authenticated in an open, interoperable and federated environment. This authentication ecosystem is depicted in the figure below.

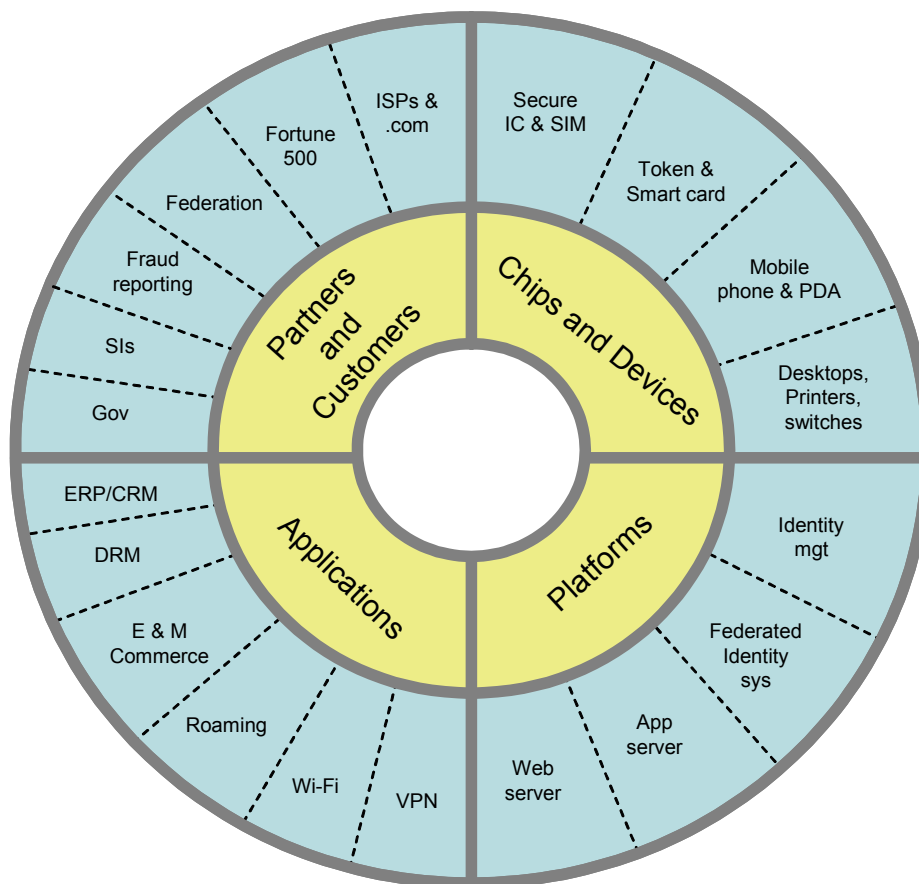


Figure 1 - Authentication ecosystem

In order to drive adoption of strong authentication across the entire user community— from corporate employees, to Internet users, to people accessing all types of resources from healthcare records to government services—the industry must collaborate to lower the complexity of, and the financial barriers to, strong authentication. Open technical standards and deployment profiles that promote interoperable

components are powerful tools for lowering complexity and cost. Therefore, the development of an open and royalty-free specification for strong authentication will be OATH's initial focus. Open, universal, strong authentication is intended to provide key constituencies (device manufacturers, identity management vendors, security service providers and application developers) with a common framework for strongly authenticating users and devices.

The open authentication vision has the following initial goals:

- To establish an open reference architecture for strong authentication, by leveraging existing open standards, where possible, and leading standardization efforts in well-established technical standards bodies where existing standards are not available.
- To propagate device credentials, strong authentication algorithms and authentication software to many network end-points (e.g. desktop computers, servers, switches, WiFi access points and set-top boxes).
- To propagate low-cost, multi-function authentication devices (e.g. tokens and smart cards).
- To transform today's mobile devices (e.g. mobile phones, PDAs and laptops) into strong authentication devices.
- To build upon well-established infrastructure components, such as directories and RADIUS servers.
- To facilitate native support (e.g. platform connectors) for strong device and user authentication on application and identity management platforms.
- To enable sharing of strong authentication tokens and credentials across organizations and networks. To leverage emerging user-centric identity technologies and federated identity protocols as powerful propagation and sharing mechanisms for strong authentication.
- To promote a vision of risk-based strong authentication where the authentication level is adjusted commensurate with the perceived risk of each transaction. To enable better evaluation of risk across organizations by enabling sharing of fraud patterns and other related information.
- To increase the number of packaged applications (e.g. enterprise resource planning (ERP), material requirements planning (MRP) and customer relationship management (CRM) applications) that support strong authentication, by providing a standard interface for management and verification of strong authentication credentials.
- To enable best-of-breed solutions through interoperable components.

To be effective, the architecture must be jointly defined and published by key industry partners that share the vision of universal strong authentication. By laying the groundwork for ubiquity, integration and interoperability, an open architecture can decrease the risk and complexity of deploying strong authentication products. In turn, the promise of reduced risk and cost will drive adoption among enterprises, service providers and governments around the World. Ultimately, by making strong authentication, for all users and all devices, part of the network fabric, the entire user community will benefit. Last but not least, by increasing trust in the network end-points, new types of secure interaction will become possible.

4. Usage Scenarios

In this section, we present some usage scenarios that highlight the need for strong authentication.

4.1. Remote Access

Salespeople, care workers, engineers and traveling executives need secure access to the corporate network while 'on the road'. These users demand the most flexible range of access methods including the following.

- VPN over wireless, whenever their laptop / PDA can connect to a WiFi hotspot
- VPN over a broadband connection from a laptop when at home
- Wireless email and other lightweight applications from a PDA or other handheld device
- Web access to email and other Web-enabled applications from an Internet café or other insecure PC

These users must be able to use a single set of secure authentication credentials at all of the access points that the enterprise has enabled.

4.2. Online Banking

These days more and more banks are making their services available via the Internet. Online banking is popular because it enables convenient 24x7 access for the bank's customers while at the same time lowers the overall transaction costs for the bank.

At the same time banks are cognizant about the increasing number and severity of threats on the Internet and are considering deploying, strong authentication, as one of the key components of a multi-pronged strategy. The key requirements for this use case are that the authentication credentials be easy to use by a diverse user population, and easy and cost-effective to deploy. Additionally, users may be required to authenticate and/or digitally sign their transactions.

In the simplest scenario banks have been using simple one-time password devices to augment traditional username-password authentication for their online commercial banking applications. Another scenario that is becoming popular in certain geographies is an EMV banking card on which separate CAP [CAP04] application has been installed. This, together with an unconnected portable reader, is capable of performing one-time-password challenge/response authentication and transaction signing (combining challenge, amount and currency). This form-factor can be used directly in the online banking channel and in 'card not present', 3dSecure transactions (where a user signs the data in an ecommerce transaction).

Since banking transactions have various levels of risk associated with them, banks are starting to deploy risk-based authentication technologies. These technologies vary the authentication technique used commensurate with the value of the transaction (e.g. viewing account balance versus transferring \$50000) and the risk associated with the request (e.g. request originating from a familiar device¹ versus a request originating from a kiosk). If the risk perceived is higher then the application should use a stronger authentication technique.

4.3. Telecommuting

In this use-case, home-based staff access their office network over a DSL or cable broadband Internet connection. The connection is typically secured using an IPsec or SSL VPN tunnel to provide the user with, essentially, the same working experience as they would have if they had been physically present in

¹ To identify the client device, the server typically calculates a 'fingerprint template' for each user's computing device (desktop or mobile) by capturing information from the device (such as a persistent cookie, the device's IP address and its browser type and version).

the corporate office. Increasingly, both computing and VoIP telephony services are being delivered to remote employees over a broadband connection, allowing telephone, email and other services to be delivered to them, as though they were present in the office.

The security of the remote location cannot be policed by the organization. So, it is critical that the user be strongly authenticated before he or she is allowed access to the corporate assets. It is essential that members of the user's family, friends and housemates be prevented from impersonating the authorized user.

4.4. Client and Business Partner Extranet

Specific individuals at client and business partner locations need to be granted deep and broad access to core business systems, typically through Web portals. They need to be securely authenticated; it is no longer sufficient to rely on just the IP address of the remote network to validate identity.

These individuals may be logging in from any Web-enabled system: a corporate desktop or home PC, for example. And so, there should be no requirement for any form of reader device to be plugged into the client system.

4.5. eGovernment

There is an increasing demand for 'joined-up' government, in which local authorities, central government departments, law enforcement agencies, healthcare providers and other agencies communicate closely, in order to provide the citizen with a more coherent and personalized service.

In order to comply with privacy regulations, it is essential that only the specific individuals at each agency, who are responsible for a citizen's case, be allowed access to their file, and this access must be independently auditable.

Secure authentication of all officials and citizens is fundamental to the secure delivery of eGovernment services.

4.6. 24x7 IT Infrastructure Support

Increasingly, SME, corporate and public-sector IT systems have to be operational 24x7, in order to support the nonstop demands of their workforces, customers and partners. IT staff and outsourced support engineers need to be able to gain instant access to core network infrastructure and servers from a remote office; from home or wherever else they happen to be when they are alerted.

The access privileges granted to some of these users give them substantial power over the organization's entire IT infrastructure. And, if any of their identities were to be stolen, then the thief would have complete control. Therefore, these users must be issued a strong form of authentication.

4.7. Wireless Roaming

Wireless users increasingly require 'anywhere access' throughout corporate offices and from public WiFi hotspots. This use case is analogous to roaming on a mobile-phone network. The end-user can seamlessly roam on a guest network. In this case, the guest network communicates with the user's home network to authenticate them. The home network could be a corporate network or a wireless service provider to which the user subscribes.

4.8. Desktop Logon

In this use case, the user is required to present strong authentication credentials in order to log into a desktop, which may be a Windows, LINUX or UNIX console. The user is typically required to present a password or a PIN. Additionally, he or she may be required to present a second factor, such as a smartcard, OTP token or a biometric sample.

4.9. Closed network

Information services, such as cable television, WiMax, etc. control access as a way of effectively charging for the services. Device authentication is used to ensure that only identified and authorized devices can receive the services of the network.

5. Authentication Framework

In this section, the high-level architecture for an open authentication system is presented. Figure 2 shows the high-level components of a generic authentication system. These components are described briefly in this section. And some of the components (viz. the client, validation, risk evaluation, provisioning and authentication and identity sharing frameworks) are discussed in greater details in Section 6.

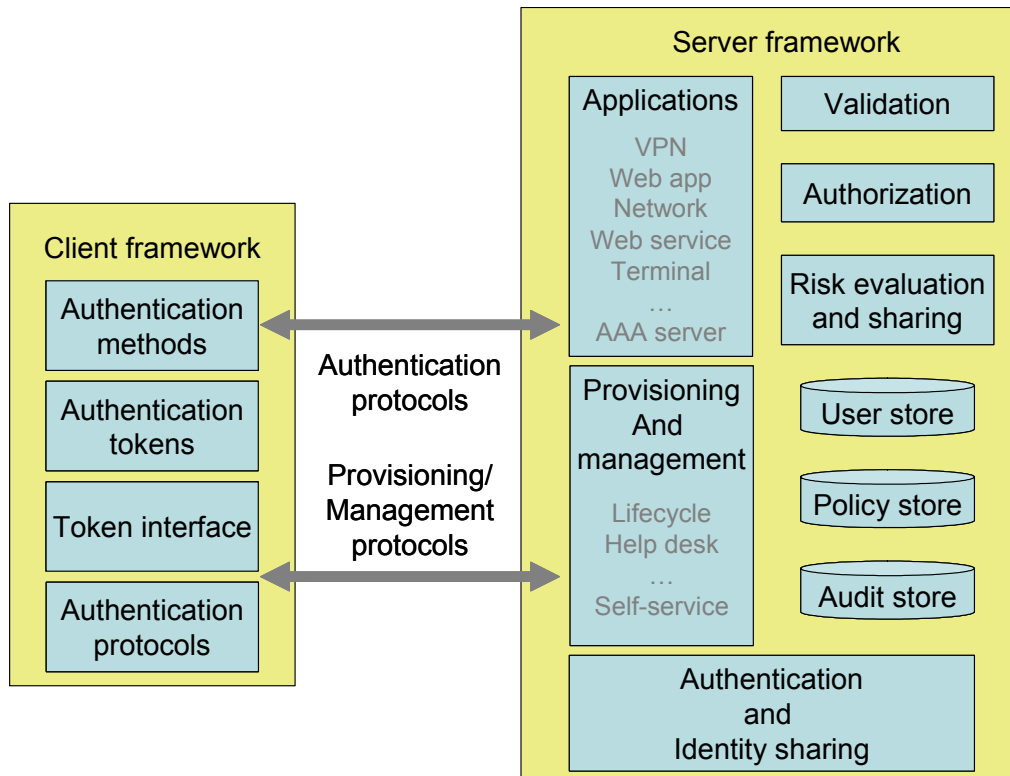


Figure 2 - Authentication architecture

5.1. Client Framework

The client framework enables a range of authentication methods, tokens and protocols to be supported when deploying strong authentication in an enterprise or service-provider infrastructure. The client framework allows standard-based integration of multiple forms of strong authentication using either existing or new authentication technologies, and communicating using standard authentication protocols.

5.2. Provisioning and Management Framework

The provisioning and management component is responsible for provisioning and managing the entire life-cycle of software modules and / or security credentials for an authentication device. The purpose of this process is to bring the device from a "clean state" to a state where it can be used as an authentication token (e.g. generating keys and provisioning an instance of an OTP token in software or provisioning a connected token). The goal of the provisioning architecture is to accommodate secure and reliable delivery of software and / or security credentials to any client device, using standard-based protocols or programmatic interfaces.

5.3. Validation Framework

The validation component is responsible for validating authentication credentials of all types. The level of assurance in the authentication event should be commensurate with the risk attached to subsequent user actions. This risk may be evaluated statically (i.e. for all actions of the same type at any time) or dynamically (i.e. taking into account contextual information, such as the user's habitual behavior or transaction pattern) by communicating with the risk evaluation and sharing module. Various applications (such as VPN gateways and Web applications) that enforce authentication policies communicate with the validation module using standard validation protocols. The goal of the validation framework is to enable system architects to deploy validation services of different types in a consistent manner throughout their systems.

5.4. Applications

Applications served by the authentication framework include those that need to strongly authenticate the end-user (e.g. VPNs, Web applications, network routers, WiFi gateways, Web services, etc.). The application communicates with the client framework using one of the standard-based authentication protocols. Once the application has received the end-user's credential, it communicates with the validation module, using one of the supported validation protocols, to validate the credential.

5.5. Authorization

Once the user has been authenticated, the application may consult an authorization module before granting the user access to the requested resources. The authorization component is responsible for determining whether the user is authorized to access a particular resource, based on the applicable access policy. The authorization component, although not used to authenticate the user, is included in this architecture for completeness.

5.6. User Store

The user store is responsible for storing all end-user profile information. This may include a unique user identifier (i.e. username), profile information (such as address, first name, last name, etc.), application-specific attributes and authentication information (such as a password or token information, etc.). The user store is typically an LDAP directory or a database.

5.7. Policy Store

The policy store is responsible for storing all policies. There are two deployment models for the policy store. The first model uses a common policy store for all components of the architecture, as shown in Figure 2. In the second model, some components (such as the validation and authorization components) may have their own policy stores.

5.8. Audit Store

The audit store is a central repository for all audit and (optionally) operational events. As with the policy store, an enterprise may choose to deploy a central audit store that collects audit events from all components, or it may allow some of the components to have their own audit stores.

5.9. Authentication and Identity Sharing

The authentication and identity-sharing component is responsible for implementing the technology primitives and models that enable sharing of authentication and/or identity information based on the models discussed below.

5.10. Risk evaluation and sharing

Risk evaluation and sharing module is used to determine the risk associated with the particular transaction. It determines the risk by comparing the transaction against known patterns for a particular user's behavior as well as by comparing it with known patterns of fraudulent activity. For e.g. based on prior known fraudulent activity a certain set of IP addresses may be deemed as high risk.

It is well known that the fraudsters typically repeat fraudulent activity across enterprises. The risk module enables sharing of fraud patterns across networks. This enables the risk module to update its fraud patterns from one or more inter-organization fraud information exchange networks and update any fraudulent patterns it has detect to one or more networks.

6. OATH Reference Architecture

6.1. Client Framework

The OATH reference architecture includes a flexible client framework for authenticating users and devices. It supports a wide range of authentication methods, tokens and protocols, for strong authentication in an enterprise or service-provider infrastructure. The client framework allows standard-based integration of multiple forms of strong authentication, implemented using either existing or new authentication technologies, and communicating using standard authentication protocols.

This section presents the high-level architecture for the OATH client framework, showing its salient features. It then identifies the existing standards / technologies and OATH focus areas for four key aspects of the client framework:

- Authentication methods
- Authentication tokens
- Token interfaces
- Authentication protocols

6.1.1. High-Level Architecture

The OATH client framework is shown in Figure 3, and the framework components and interfaces are described below.

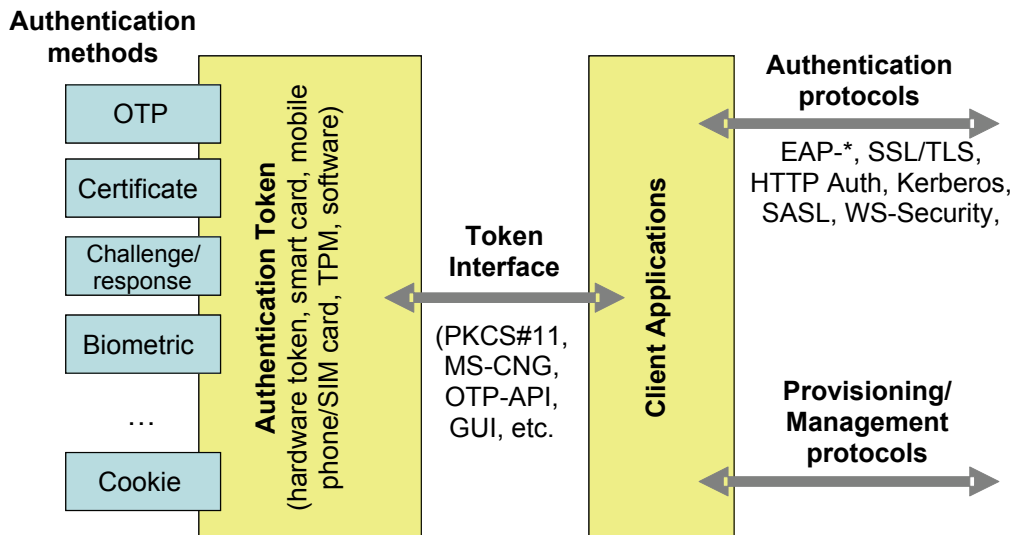


Figure 3 – Client framework

Authentication method - A function for authenticating users or devices, including one-time-password (OTP), symmetric key challenge / response, public key certificates and other methods.

Authentication token - A hardware or software container that implements one or more authentication methods, and performs the security-critical client-side authentication operations and secure storage of authentication credentials.

Token interface - The interface by which an authentication token communicates authentication data and credentials with the client application. This interface may be either an API, in the case of a token that is connected to the client application or a GUI, in the case of a standalone token, such as an OTP token.

Authentication protocol – The over-the-wire protocol used to exchange authentication data between the client application and the server application. Authentication protocols support one or more authentication methods, and there are many existing industry standard protocols.

Provisioning / management protocol - The protocol used to provision authentication credentials and authentication token software, as well as providing ongoing life-cycle management. The OATH reference architecture does not address user / identity provisioning, which is covered by other standards organizations. The provisioning and management functions are addressed separately in Section 6.4.

Client Application - The software client that uses strong authentication to control access to an application function, either locally or remotely over a network connection (such as the Web, VPN or email), utilizing an authentication protocol to exchange authentication data with remote server applications. The client application includes a provisioning agent to support client provisioning and management, as defined in Section 6.4. The client application supports one or more token interfaces, and typically runs on a PC or mobile device owned by a consumer or enterprise user.

6.1.2. Salient Features

The OATH client framework includes the following key features, to allow flexible strong authentication deployments that offer cost, security, user convenience, functionality and performance characteristics that suit the application.

User and device – Supports authentication of both users and devices.

Standalone and embedded - Supports both standalone token devices and embedded token / integrated-device implementations.

Connected and unconnected - Supports both connected and unconnected modes of token operation.

Multi-vendor interoperability - Supports standard authentication algorithms to enable multi-vendor interoperability of authentication tokens and validation servers.

Multi-key - Supports multi-key tokens (e.g. multiple instances of OTP credentials on one device, each having a unique token ID associated with its service provider).

Multi-function - Supports multi-function authentication tokens (i.e. multiple authentication methods supported on a single device, such as OTP and certificate-based authentication on the same smart card).

Multi-factor - Supports multi-factor authentication (i.e. two or three factor authentication as part of a single session negotiation).

Wired and wireless - Supports Internet-based client applications and mobile / wireless client applications for strong authentication from anywhere and from any device.

Multiple token interfaces - Supports multiple token interface APIs: PKCS #11, MSCNG and others in the future.

Multiple clients - Supports multiple client applications (potentially implementing different authentication protocols) with a common authentication token.

Trusted proxy - Supports a trusted proxy server implementation of the client authentication token function to support low-end mass-market client devices (e.g. server-side OTP generation with SMS transmission to mobile phones).

6.1.3. Authentication Methods

Authentication methods describe how to authenticate individual users and / or devices. Some authentication methods and their relationships to authentication protocols are described below.

Existing Standards and Technologies

Password – The password authentication method is the oldest and still most commonly used method for user authentication. However, password authentication is no longer considered adequately secure in many applications, because users can share their passwords and because replay attacks have become common. Password authentication, on its own, is not recommended by OATH. But, in order to provide a more complete picture of available methods, it is described here. The password authentication method is based on data matching. The user is identified by a user ID and their authentication password is matched with the corresponding data stored by the validation service for that user.

One-Time-Password – The one-time-password, or OTP, authentication method can be divided into two sub-types. Time-based methods rely on the transformation of a shared secret and a time value that is synchronized between the server and the client. Event-based methods rely on the transformation of a shared secret and an event count that is synchronized between the server and the client. Typically, the event that is counted is the pressing of a button on the token. [HOTP]

Challenge / Response – The challenge / response authentication method is usually based on a shared-secret transformation using symmetric-key hashing techniques. The server side sends the client a challenge, and the client uses this challenge and the shared secret as input to the transformation. The resulting output is the response, which is then sent to the server.

Transaction Signing – This method is a more advanced version of challenge / response, where the user confirms certain details of the transaction (e.g. target account number, amount and currency, for a funds transfer transaction). These details are then input into the algorithmic computation, often based on symmetric cryptography. EMV-CAP Mode 2 with TDS [CAP04], APACS prompted Data Signing or OCRA are examples of algorithms that support transaction signing. Usually the server would transmit the specific details and the user would type them into a token or an unconnected smart card reader. The token / reader would then display a response that is sent to the server for verification.

User certificate – Certificate-based authentication uses public-key encryption techniques, supported by a public-key infrastructure (PKI) for key and certificate management. Digital certificates are issued by a certification authority and they bind the user's identity to their public key. In a typical certificate-based authentication protocol, the client uses its private key to sign a challenge from the server, and the server verifies the signature using the client's certificate. [PKIX]

Biometric - Biometric authentication methods are based on a physiological characteristic of a user, such as a fingerprint, iris image or facial image. Biometric authentication represents the "what you are" component of multi-factor authentication. Biometric authentication is based on data-matching of the captured biometric characteristic to a stored template.

Device fingerprint – A Web application may examine a persistent cookie, the source IP address and the type and version of the remote user agent. This information can be used to identify suspected impersonation attacks. However, because users legitimately change or update their browsers periodically, this technique is subject to "false positives". Nevertheless, it can be used as one in a set of risk metrics to decide when step-up authentication is required.

Device certificate – Certificates can be embedded in a variety of network devices, such as cable modems, set-top boxes and WiMax-compliant subscriber stations.

Trusted platform module – Trusted platform modules can be used to strongly authenticate appropriately equipped computing devices. [TPM]

OATH Focus Areas

OATH has endorsed a new OTP algorithm standard called HMAC-based OTP [HOTP], based on the HMAC SHA1 algorithm. It is an event-based OTP algorithm, in which a counter value is used in the OTP calculation and incremented on both the client and server after each use. The algorithm has been approved by the IETF for standardization as an Informational RFC [RFC4226]. OATH has also developed a challenge / response algorithm based on HOTP, called OCRA (OATH Challenge-Response Algorithm) and has submitted this specification to the IETF as an Internet Draft. OCRA also supports short digital signatures. It will be submitted to the IETF for publication as an RFC in 2007. A time-based OTP algorithm variant based on HOTP is also being developed as a 2007 roadmap item.

Areas of future work include additional extensions to the current HOTP algorithm.

- Develop a counter-based resynchronization method for clients that can send the count value to the server along with the OTP value
- Develop a composite shared-secret (e.g. based on a user's PIN or other deterministic data, for computing the shared secret)
- Add a data field for computing OTP values

Additionally, OATH will promote standardization of other low-cost authentication technologies, specifically ones that address consumer usage scenarios.

Some of the areas that OATH is investigating include scratch cards and methods derived from battleship or bingo cards².

6.1.4. Authentication Tokens

The OATH client framework is designed to support existing hardware token implementations, as well as soft-token implementations on PCs, mobile phones and PDAs, and new technologies that combine multiple authentication methods on a single token. Typical authentication system deployments support multiple types of tokens to cover a broad range of user profiles, security requirements and application scenarios. By using standard token interfaces and standard authentication algorithms, such as the [HOTP], a range of authentication tokens can be supported in a uniform way within the OATH reference architecture.

Existing Token Types

There are many forms of authentication token deployed today, including:

- Standalone OTP generators
- Smart cards
- EMV payment or debit cards containing a separate CAP application, in conjunction with an unconnected reader
- USB key fobs
- Software tokens

² In a battleship or bingo card, each user has a card printed with a unique two-dimensional grid of characters. During authentication, the application challenges the user to enter characters that are located at a dynamically-chosen set of coordinates in the grid.

-
- Trusted platform modules [TPM]

OATH Focus Areas

OATH's objective is to foster innovation by encouraging the embedding of strong authentication technologies into devices that the user carries for another purpose, such as a mobile phone, and to provide flexibility and reduced cost through multi-function tokens. To allow for innovative token solutions, OATH intends to champion the development of standards that are easily implemented on a range of token types, as in the case of the HOTP algorithm. Specific areas of focus for tokens include the following.

- Define soft tokens on mobile devices
- Define SIM-based authentication tokens
- Develop multi-key tokens to access multiple services using the same physical token
- Develop a reference implementation of the HOTP algorithm, to accelerate adoption (e.g. Java smart card)
- Develop a reference implementation of OCRA on a token
- Develop a profile for an OATH identifier namespace to identify OATH credentials in a standard vendor-independent format, based on the IEEE EUI-64 standard [EUI64]

6.1.5. Token Interface

The OATH reference architecture includes industry-standard interfaces between the client application and the authentication token to enable interoperability between different vendors' tokens and client applications. The client framework also allows the authentication token to be integrated directly into the application, as is the case for a soft token or SIM card on a mobile device, or to be operated as a standalone device in either connected or unconnected mode. In connected mode, the token device may interface to the client application either via a cryptographic API, such as PKCS #11 or MSCNG, or via an API dedicated to the particular authentication method. In unconnected mode, the token requires a user interface to display authentication values, such as one-time-passwords, and, optionally, a keypad to enter a PIN, password or transaction details. Token Interfaces may support either single function or multi-function tokens (i.e. those that support more than one authentication method, such as OTP and certificate-based authentication, on a USB token).

Existing Standards and Technologies

Token interface standards already exist for PKI-based authentication, specifically:

PKCS #11 – See [PKCS#11].

MSCNG - Microsoft Cryptographic API: Next Generation [MSCNG] for Windows.

Other authentication methods rely on proprietary token interfaces.

OATH Focus Areas

The OATH reference architecture envisions both extensions to existing industry standard APIs and the development of new APIs for the authentication token interface.

- Develop extensions to existing APIs to support OTP algorithms
- MSCNG will require extensions to support HOTP
- Other device-specific API extensions will be required as HOTP gains adoption
- New token interface APIs are foreseen
- A platform-agnostic thin API standard for the OTP token interface is required to allow multi-vendor interoperability of hard / soft tokens and client applications, while insulating the application developer from platform-specific implementations (the OTP retrieval API may be documented

separately from the token provisioning / management API, since many application developers will not require access to the token provisioning and life-cycle management functions)

- HTML tags will be defined to identify authentication data elements in Web applications

6.1.6. Authentication Protocols

Over-the-wire authentication protocols are used to exchange authentication data between the client and the server. Each authentication protocol supports one or more authentication methods. The OATH reference architecture accommodates existing protocols, and envisions the development of extensible protocols that can support new authentication methods as they are defined.

Existing Standards and Technologies

Providing a comprehensive framework for authentication services requires that we first have an understanding of protocols and mechanisms currently in use. Listed below are the most common authentication protocols with short descriptions of their use and references to more detailed descriptions.

Challenge Handshake Authentication Protocol - CHAP is an authentication protocol used to log a user on to an Internet access provider. It was widely used in early dialup services. See [RFC1334] and [RFC1994].

Extensible Authentication Protocol - EAP is used between a dialup client and a server to determine what authentication protocol will be used. EAP is also widely used for other client / server authentication services. See [RFC3748].

Generic Security Service Application Program Interface – GSSAPI provides security services to calling applications in a generic fashion, supported by a range of underlying mechanisms and technologies. It allows source-level portability of applications to different environments. The authentication method specified by GSSAPI is very generic and further defined in other RFCs that build on GSSAPI. See [RFC1508].

HTTP Basic and Digest Authentication - HTTP authentication describes username / password authentication for HTTP 1.1. It is commonly used in combination with SSL to provide confidentiality for the password (Basic) or a cryptographic hash of the password (Digest) as it is sent over the channel. See [RFC2617]

Kerberos – The Kerberos network authentication protocol is used in a distributed computing environment. It is based on the principle that the user authenticates to an authentication server. The server then grants the user the right to request tickets from one or more ticket granting servers that issue authentication tickets for any application that the user has the right to access. Kerberos acts much like a single-sign-on solution between applications and trusted computers. Kerberos is, for example, used in Microsoft Windows 2000 and above. See [RFC1510].

MSCHAP v1 and v2 - Microsoft's PPP CHAP dialect (MSCHAP) extends the user authentication functionality provided on Windows networks to remote workstations. MSCHAP is derived from the PPP Challenge Handshake Authentication Protocol. See [RFC2433] and [RFC2759].

Password Authentication Protocol – PAP is a two-way handshake protocol designed for use with PPP. PAP is a plain-text password protocol used on older SLIP systems. It is not considered secure, because it passes the credentials in clear text. See [RFC1334] and [RFC1994].

Simple Authentication and Security Layer - SASL defines a method for adding authentication support to connection-based protocols. See [RFC2222].

S/Key – S/Key is an early one-time-password system, based on hashing, that protects against password replay. See [RFC1760] and [RFC2289].

SSL/TLS - The transport layer security protocol is widely supported in standard Internet browsers and Web servers. It is based on digital certificates, and can provide mutual authentication. See [RFC2246].

WS Security – The OASIS Web Services Security specification describes enhancements to SOAP messaging that protect message integrity, confidentiality and source authentication. These mechanisms can be used with a wide variety of security models and encryption technologies. WS-Security also provides a general-purpose mechanism for associating security tokens with messages. See [WSS].

Sideband signaling - In certain types of man-in-the-middle and Trojan attacks, the adversary impersonates the user by using the user's genuine credentials on the user's own device. Sideband signaling may be used to mitigate this threat. Sideband signaling can be directly integrated into the authentication protocol. This is a particularly appealing approach in situations where wireless devices and wired devices are both participating in the authentication process, thereby providing multiple IP communications paths to the user's location. In addition to direct integration into the authentication protocols, sideband signaling may be used to select the authentication protocol to be used. Sideband signaling can further mitigate these attacks by requiring explicit out-of-band notification and / or confirmation of high-value transactions.

OATH Focus Areas

OATH's focus is to ensure that OATH credentials are supported as first class citizens in the various authentication protocols that exist today.

6.2. Validation Framework

In any large organization, there may be several applications that make use of strong authentication. But, it is unlikely that the same authentication method will satisfy all application and user constituencies. Today, such an organization must deploy multiple authentication methods and possibly separate infrastructures to support those methods.

The OATH validation framework enables vendors to write custom validation modules and it enables enterprises to deploy multiple types of authenticators in the same infrastructure. Additionally, the validation framework enables organizations to deploy multiple protocol handlers, such as RADIUS, OCSP, etc.

The OATH validation framework benefits the organization that deploys strong authentication by enabling the use of multiple authentication methods in the same infrastructure, thereby enabling phased rollout of strong authentication solutions across a wider set of applications and user groups.

Today, each authentication vendor has to build a complete validation server. Implementations of the OATH validation framework will remove the necessity for a vendor to rebuild a complete solution. Some vendors may specialize in validation servers that comply with this framework. Others that offer identity management products, such as AAA servers, may choose to implement this framework within their existing products. Other vendors may choose to provide hosted validation services based on this framework. This framework also enables vendors to develop pluggable modules that are specific to the particular authentication method that they offer. The framework enables vendors to bring innovative solutions to market more quickly.

6.2.1. High-Level Architecture

As shown in Figure 4, the validation framework consists of the following sub-modules.

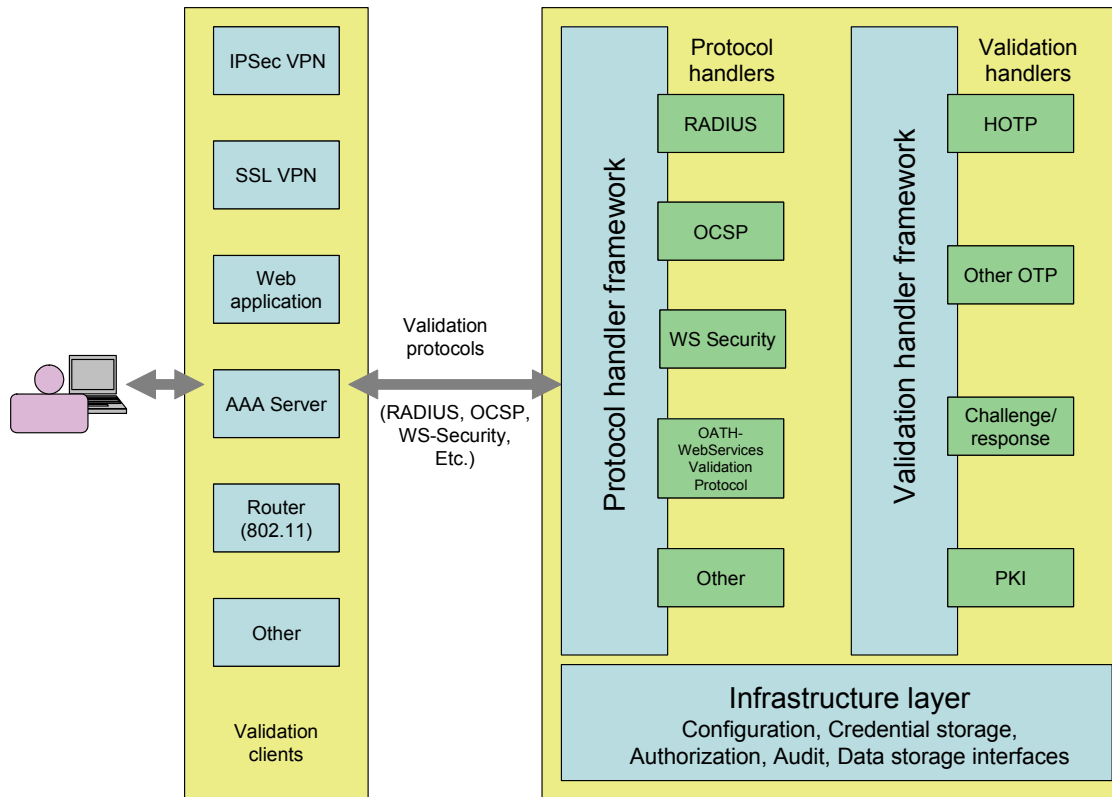


Figure 4 - Validation framework

Protocol handler framework - This framework enables deployment of multiple protocol handlers. The framework is responsible for I/O, threading, loading the handlers, etc. The framework defines the necessary interfaces and configuration parameters.

Protocol handlers - A protocol handler is the component that implements support for a particular validation protocol, such as RADIUS, OCSP or the OATH-WebServicesValidation Protocol.

Validation handler framework - This framework enables deployment of multiple validation handlers. Among other things, the framework defines the necessary interfaces and configuration parameters and is responsible for loading validation handlers.

Validation handlers - Each validation handler supports validation of a particular authentication method (e.g., HOTP, RSA SecurID, etc.).

Infrastructure layer - The infrastructure layer provides some of the common functions that can be used by the other components in the validation system. These include the various protocol and validation handlers, as well as the frameworks themselves.

Configuration module - This module allows the registration and configuration of the various entities in the system, such as the validation clients, protocol handlers and validation handlers.

Authorization module - This module configures the access policies that control which validation clients can access which validation handlers.

Audit / logging module - This module allows the logging of various audit and operational events by the components of the validation framework.

Data store interfaces - The data store interfaces are listed below. These interfaces enable the other system components to store data, independent of the specific data storage technology.

- Configuration store interface
- Token store interface
- Audit / log store interface

6.2.2. Salient Features

The salient features of the OATH validation framework are as follows.

Multiple authentication methods - The validation framework supports validation of multiple authentication methods, such as certificates, OTPs, challenge / response, etc., simultaneously. The framework enables organizations to change the supported authentication methods by adding or removing validation handlers. Additionally, the validation framework enables an enterprise to deploy multiple flavors of a particular authentication method that can, for example, validate OTPs from different vendors (e.g. HOTP and EMV-CAP), or validate certificates from different certification authorities. The validation framework also has the ability to validate more than one authentication credential in the same transaction.

Multiple validation protocols - The validation framework enables validation clients to submit validation requests using a variety of protocols, such as RADIUS, CRLs retrieved via HTTP or LDAP, OCSP, OATH-WebServiceValidationProtocol, etc., simultaneously. The validation framework enables organizations to change the supported protocols by adding or removing protocol handlers³. The framework also enables a deployment with multiple instances of the same validation protocol.

Multiple validation clients - The validation framework enables multiple applications that need to validate credentials (validation clients) to send requests simultaneously. The framework enables administrators to configure authorization policies that control which validation clients can access which validation handlers.

Risk adaptive – The ability to enforce authentication methods that match the level of assurance to the risk of identity fraud.

Standardized configuration - The validation framework supports standard primitives to configure the validation and protocol handlers. It also provides a standard way to register and query the handlers.

Framework for logging audit and operational events - The validation framework supports common methods for the handlers, and the framework components, to log audit and operational events.

Deployment agnostic - The validation framework may be deployed either on the premises of the organization, or as a hosted service. Additionally, validation handlers may make remote calls to distributed components in order to perform validation.

Data store abstraction - The validation framework provides appropriate data storage interfaces for data such as token data, audit and operational data and configuration data. This allows an organization to use the data storage technology that best meets its requirements.

³ Note that not all protocol handlers are able to support all the different authentication methods. I.e. not all the paths of the 'imaginary crossbar switch' can be active. There are a couple of reasons for this – one is that some protocols are designed for specific credential types (e.g., OCSP for PKI), and, secondly, it may depend on the specific mode being used for the protocol, such as RADIUS (e.g., RADIUS CHAP).

6.2.3. Existing Standards and Technologies

Existing Validation Protocols

Lightweight Directory Access Protocol - LDAP is a protocol for accessing on-line directory services. LDAP defines a relatively simple protocol for updating and searching directories over TCP/IP. Directories are used to store information about end-users, including usernames and passwords. Consequently, LDAP is often used by applications to validate the username and password of an end-user.

Online Certificate Status Protocol - OCSP is a method for determining the revocation status of an X.509 digital certificate without directly using a CRL. OCSP's request / response nature leads to OCSP servers being called OCSP responders. See [RFC2560].

Remote Authentication Dial In User Service - RADIUS is an authentication, authorization and accounting (AAA) protocol for applications such as network access, which can be used in both local and roaming situations. See [RFC2138].

Diameter - Similar to RADIUS, Diameter is an authentication, authorization and accounting (AAA) protocol for applications such as network access or IP mobility. See [RFC3539].

Terminal Access Controller Access-Control System - TACACS is a remote authentication protocol. It is commonly used in UNIX networks. It allows a remote access server to communicate with an authentication server in order to determine if the user should be allowed access to the network. See [RFC1492].

XML Key Management Specification - XKMS provides a secure method for registration and subsequent life-cycle management of public-key information. The XML Key Management Specification (XKMS) comprises two parts: the XML Key Information Service Specification (XKISS) and the XML Key Registration Service Specification (XKRSS). XKISS may be used to validate a signature and the certificate associated with that signature. See [XKMS].

Existing Authentication and Validation Interfaces

Pluggable Authentication Modules - PAM is a generalized API for authentication-related services, which allows a system administrator to add new authentication methods simply by installing new PAM modules, and to modify authentication policies by editing configuration files. PAM was first developed by Sun Microsystems and is currently supported in Solaris, Linux, FreeBSD and NetBSD. See [PAM].

Java Authentication and Authorization Service - JAAS is an API that enables Java applications to access authentication and access-control services without being tied to those services. It implements a Java technology version of the standard Pluggable Authentication Module (PAM) framework, and supports user-based authorization. This permits Java applications to remain independent of the underlying authentication technologies. New or updated technologies can be plugged in without requiring modifications to the application itself. Several sample authentication modules are available that implement JNDI (Java Naming and Directory Interface), UNIX, Kerberos and Windows NT authentication. See [JAAS].

Existing Protocol Handler Frameworks

Multipurpose Infrastructure for Network Applications - MINA is a framework for building network applications with minimal effort. It provides a framework into which protocol providers (such as LDAP, DNS, etc.) can be plugged, without having to deal with low-level I/O semantics and issues such as concurrency. MINA provides support for both blocking and non-blocking network I/O, using the Java NIO libraries. See [MINA].

6.2.4. OATH Focus Areas

OATH has chosen the following focus areas.

- OATH intends to promote the development of appropriate interfaces for both protocol and validation handlers that will enable vendors to write pluggable handlers, as described above. In particular, it will define JAAS modules for the HOTP and OCRA authentication methods.
- Existing validation protocols may be used by applications to authenticate an end-user's credentials. However, these protocols may not be adequate to support all the envisioned authentication methods, credential types and deployment topologies (enterprise hosted or outsourced). OATH will evaluate the requirement for standardized extensions to existing protocols (e.g. RADIUS), and will investigate whether a requirement exists for one or more additional validation protocols to meet the requirements of strong authentication.
- OATH will develop a Web-services validation protocol specification suitable for a variety of authentication methods.
- OATH will develop a Web service interface for sharing identity-fraud pattern data.

6.3. Risk evaluation and sharing framework

Many organizations are starting to augment their deployments of strong authentication technologies with a risk-based approach. The principal goal of the approach is to tailor the strength of authentication to the level of risk associated with the actions requested by the user.

This approach is also typically more user-friendly. By using this approach the user can use more convenient⁴ authentication methods in low-risk situations and needs to use the stronger authentication technology when the risk is perceived to be high, for e.g. when accessing the application from an unknown terminal or when transferring a large amount of money.

Another benefit of this approach is that it reduces the 'false-positive' events – i.e. the scenario when a genuine user is denied access to the application.

Risk can be calculated in a number of different ways. One approach is comparing the user's current behavior with their habitual behavior pattern – for example, a bank may have a customer who is a housewife and typically accesses the application from the home computer. So, if there is an access from another computer that could be perceived as an anomaly. Another approach is comparing the user's current behavior with known patterns of fraudulent behavior (e.g. request originating from known high risk country), from the level of assurance in the user's local network, etc.

The rest of this section describes a framework for incorporating risk-based authentication into applications.

6.3.1. High-level architecture

Figure 5 shows the high level architecture for the *risk evaluation and sharing* framework. .

⁴ There is typically an inconvenience cost associated with any strong authentication technology – maybe this is because the user has to carry a token, or maybe because the user has to download special purpose software on their machine.

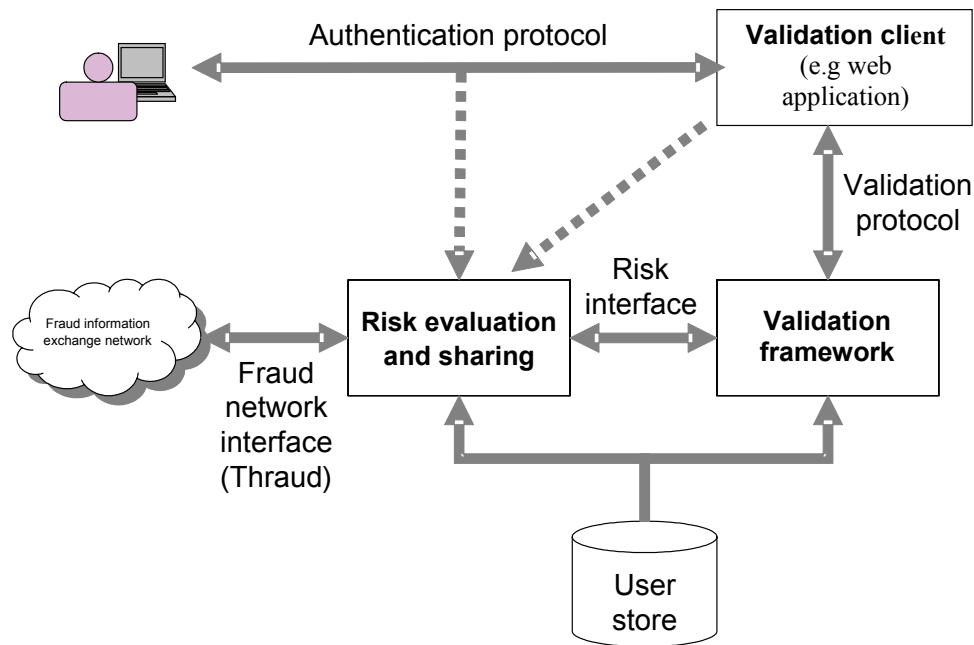


Figure 5 - Risk-based authentication architecture

The validation framework or the application (i.e. validation client) directly, queries the risk-evaluation and sharing module using the risk interface. The risk component returns a risk score associated with that particular transaction or request. Additionally, it may also return a set of authentication method identifiers corresponding to the authentication methods that are suitable for that level of risk.

The validation framework or the validation client can then request the user to authenticate using one of the suitable authentication methods. If the user successfully presents an appropriate authentication credential then the risk is mitigated to an acceptable level.

As shown in the figure above, the risk evaluation and sharing component can learn the necessary information about the user request in one of two ways:

1. It can monitor the user message-flow directly, or
2. The information can be supplied via the risk interface.

The risk component may also query the user store to find out what authentication mechanisms are available to the user in question.

Finally, the framework also enables sharing of fraud patterns by connecting with the *fraud information exchange network*. The network enables the risk module to update its database of fraud patterns from the broader community while at the same time upload information about fraudulent transactions and patterns that it has detected.

6.3.2. Salient features

The salient features of the OATH risk framework are as follows.

Open network interface – leverages experience across the industry

Step-up authentication – responds to the calculated risk

Personalized authentication methods – selects only authentication methods available to the subject

6.3.3. Existing Standards and Technologies

FS-ISAC - Anonymous reporting of incidents through trusted intermediaries, such as FS-ISAC

BITS - BITS and its Partner Group are looking at sharing a 'negative database', containing lists of closed, bad or suspect accounts and players (e.g. bad merchants and individuals), such as is being built by Early Warning, MasterCard, VISA and others

eFunds – Fraud detection capabilities in Checks and Cards, Merchant networks

RSA - eFraudnetwork sharing information and suspicious patterns

Bharosa - Proactive Fraud Intelligence Net sharing risk models for on-line fraud and fraud patterns

LUCID – Letix Universal Case Identification Database for law enforcement and other government information-sharing initiatives

6.3.4. OATH Focus Areas

Work items for much of this architecture have already been defined or already exist on the OATH roadmap. However, the following additional work items have to be completed.

- **Fraud-pattern exchange** - OATH will complete development of a Web service interface specification for exchanging identity-fraud pattern data. OATH has currently proposed a draft standard for exchanging transaction fraud information [THRD].
- **Risk interface** - The risk interface has to be defined. It allows the validation framework (or application) to pass relevant information about the request/transaction and obtain, in return, a risk score and/or identifiers for the set of authentication methods that both satisfy the risk requirement and are available to the user.
- **Validation protocol extension point** - Extension points must be added to both the validation protocol (OATH-WebServicesValidationProtocol) and the risk interface to allow the application to pass the necessary context information to the risk component (schema for the context information will not be defined at this stage).
- **Language bindings** - Bindings for both Java and Web-services are required for the risk interface.
- **Database schema** - The need to define database schema to contain the identifiers for the authentication methods available to the user will be explored.

6.4. Provisioning and Management Framework

Devices come in many shapes and sizes, and their capabilities and functionality vary widely. It is challenging to implement a single protocol for provisioning, and the subsequent life-cycle management, of software components, for all the different types of security credentials and devices that are available. The goal of the OATH provisioning and management architecture is to specify a framework that can accommodate multiple standard-based provisioning protocols, in order to provision different types of credentials to all types of devices. Additionally, for some platforms, the framework will enable provisioning of authentication software to the device.

Typically, software components can be provisioned to a device using one of two methods:

1. Embedded at the time of manufacture or personalization of the device, e.g. SIM cards and hardware tokens.

2. Loaded to the device, post-personalization, over a network interface, e.g. PDAs and mobile phones.

The same options exist for provisioning security credentials.

Security credentials may also be provisioned to a secure repository and made available to a trusted proxy server, such as an SMS OTP proxy server, in order to generate OTP values on behalf of the device.

The OATH provisioning architecture offers a generic and extensible framework for provisioning the authentication token software modules and / or associated security credentials to devices.

6.4.1. High-Level Architecture

Figure 6 illustrates the OATH provisioning and management framework.

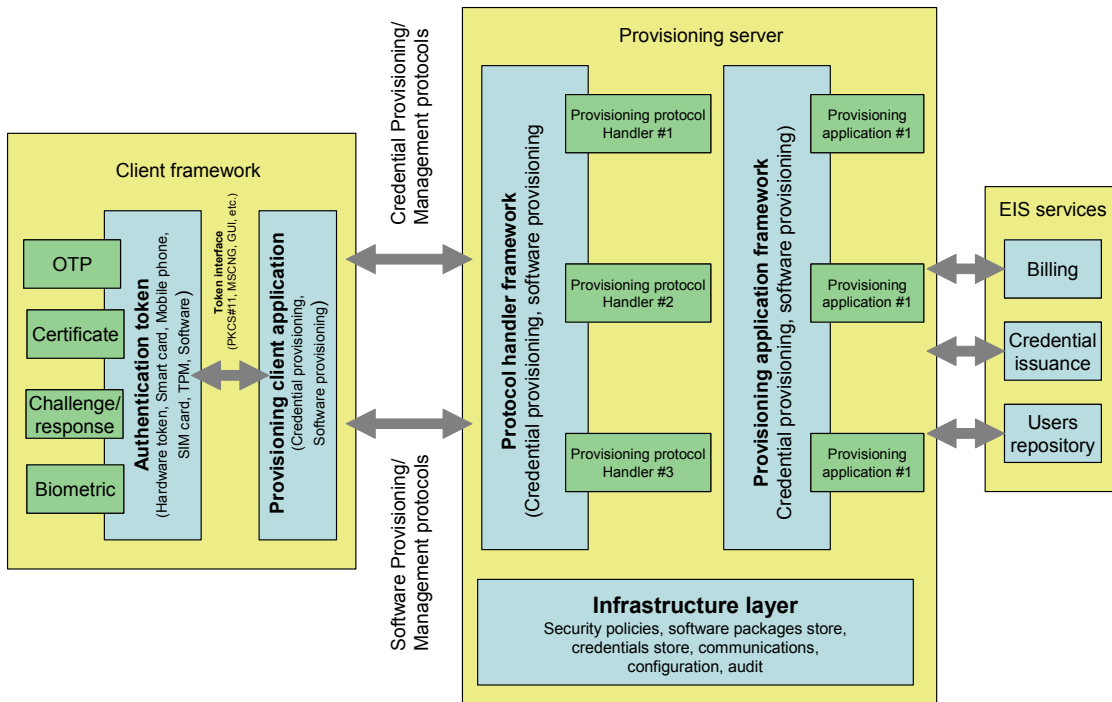


Figure 6 - Provisioning framework

As shown in the figure, the provisioning architecture consists of three tiers:

- The provisioning client application
- The provisioning server
- The enterprise information systems or services

The Provisioning Client Application

The provisioning client application runs on a device or a personal computer. The provisioning client application is responsible for provisioning the authentication token software modules and / or associated security credentials to the device. The provisioning client application may implement one or more provisioning protocols and may support one or more types of credential. Multiple provisioning client applications may also coexist on the same device.

The Provisioning Server

As shown in the figure, the provisioning server consists of the following components.

Protocol handler framework - The protocol handler framework is built on top of the infrastructure layer and supports the provisioning application framework through a generic application programming interface. Different device platforms require different provisioning protocols. The framework enables deployment of multiple provisioning protocols by specifying the necessary interfaces and configuration parameters. The protocol handler framework may host several protocol handlers. These may include credential-specific protocol handlers, such as XKMS, CTKIP and DSKPP, or software-specific protocol handlers, such as J2ME DL and OMA DLOTA.

Provisioning application framework - The provisioning application framework defines generic interfaces and configuration parameters for building and deploying provisioning applications. A provisioning application implements the business rules for processing provisioning requests for credentials and / or software. The provisioning application may process provisioning requests internally or delegate them to an external system, such as a credential issuer, for servicing. Multiple provisioning applications can coexist within the provisioning application framework. For example, one application may be responsible for provisioning and managing the life-cycles of security credentials, such as HOTP secrets, while another application may be responsible for provisioning and managing updates of the authentication token software modules on the devices.

Infrastructure layer - The infrastructure layer forms the basis for the provisioning system. It provides some of the common functions, such as client authentication, configuration, persistent storage and communication to the top-level components of the system (i.e. the provisioning application framework and the protocol handler framework). The infrastructure layer services may be provided by several modules, including the following.

- **Customer management module** - This module allows access to, and management of, customer records.
- **Device inventory module** - This module performs the registration and configuration of supported devices, such as PDAs, mobile phones and SIM cards.
- **Configuration module** - This module performs the registration and configuration of the components of the system, such as the protocol handlers, provisioning applications and interfaces to external enterprise information systems and services.
- **Security policies module** - This module performs the configuration of the access policies that control which provisioning clients may access which protocol handlers.
- **Persistent storage interfaces** - These interfaces enable the system components to store and / or access persistent data, such as configuration data, credentials, software packages, security policies, business rules and audit data.
- **Enterprise Information Systems / Services** - The provisioning application may connect through the infrastructure layer to various external enterprise information systems and services, in order to fulfill the business requirements. Such external systems may include:
 - Credentials issuer
 - Billing / payment system
 - Enterprise user directory

6.4.2. Salient Features

The OATH provisioning framework supports the following key features.

Supports existing standards - Implementations of the OATH provisioning architecture allow vendors to implement existing standard-based provisioning protocols. The framework also enables vendors and customers to deploy proprietary provisioning protocols for provisioning and managing specific credentials and devices.

Credentials provisioning - The protocol handler framework enables implementation and deployment of secure protocols for credentials provisioning and subsequent life-cycle management (renewal, revocation, suspension and reactivation).

Software provisioning - The protocol handler framework enables implementation of secure protocols for software provisioning and subsequent life-cycle management (i.e. update and uninstall).

Multiple credential types - The protocol handler framework allows multiple credential provisioning protocols to coexist and, hence, it allows provisioning of different types of credentials (shared keys, certificates, etc.) to various devices.

Multiple device types - The framework supports a wide range of devices types.

Optimized provisioning protocols - The protocol handler framework supports highly optimized and customized provisioning protocols to cope with device constraints, such as network latency, bandwidth, memory, cryptographic capacity and processing power.

Multiple provisioning client applications - The client framework enables multiple provisioning client applications to coexist, thereby handling different types of credentials on the same device.

Standardized configuration - The provisioning framework supports standard primitives for configuring, registering and querying the protocol handlers and provisioning applications.

Framework for logging audit and operational events - The provisioning framework supports common methods for protocol handlers and provisioning applications to log audit and operational events.

Data store abstraction - The provisioning framework provides appropriate data storage interfaces for data, such as credential data, audit and operational log data and configuration data. This allows an organization to use whatever data storage technology best meets its requirements.

6.4.3. Existing Standards and Technologies

There are several methods for secure credential provisioning to connected devices. However, existing methods and protocols are usually designed to support one type of credential only. The objective of the OATH provisioning framework is to support credential issuance, provisioning and other life-cycle management functions for all types of credentials (e.g. symmetric keys, asymmetric key-pairs, certificates) and for all types of devices.

Existing Credential Provisioning Protocols

This section describes some of the credential provisioning and related security protocols that can be leveraged by the OATH provisioning architecture.

XML Key Management Specification - XKMS provides a secure method for the registration and subsequent life-cycle management of public-key information. It comprises two parts: the XML Key Information Service Specification (XKISS) and the XML Key Registration Service Specification (XKRSS). The XKRSS specification defines an interface for a Web service that performs registration of public-key information. Once registered, the public key may be used in conjunction with other Web services, including XKISS. See [XKMS].

PKCS #10, CRMF and PKCS #7 - PKCS #10 and CRMF define standard syntax for certificate requests. Certificate requests are sent to a certification authority, which transforms the request into an X.509 public-key certificate. The resulting certificate, or certificate chain, is usually returned in PKCS #7 format. PKCS #10, CRMF and PKCS #7 are widely supported in public-key infrastructures and public-key enabled applications. See [RFC2986], [RFC2511] and [RFC2315].

Certificate Management Protocol - CMP provides protocols for certificate requests and management. The protocol messages are defined for X.509 certificate creation and management. CMP defines online interactions between PKI components, including an exchange between a certification authority and a client. See [RFC2510].

Certificate Management Messages over CMS - CMC is a certificate management protocol using CMS. The protocol defines an interface to public-key infrastructure components, based on CMS, PKCS #10 and CRMF. See [RFC2797] and [RFC2630].

Simple Certificate Enrollment Protocol - SCEP was proposed by Cisco in an Internet draft. Its most compelling feature is the possibility for automatically enrolling certificates for large-scale installations. SCEP supports the RSA public-key algorithm only and leverages PKCS #7. See [SCEP].

Simple Password-authentication Exponential Key Exchange – SPEKE provides authentication and key establishment over an insecure channel using only a small password, without risk of off-line dictionary attack. SPEKE is a variant of Diffie-Hellman Encrypted Key Exchange (DHEKE). See [SPEKE] and [DHEKE].

Crypto Token Key Initialization Protocols (CTKIP) Proposal - The CTKIP proposal provides a secure method of initializing and configuring cryptographic tokens with secret keys, without exposing the secrets to any entities other than the server and the cryptographic token itself. The protocol does not require private-key capabilities in the cryptographic token, and does not mandate an established public-key infrastructure. The initialization session may be secured either using a key, agreed beforehand between the client and the server, or using the server's public key. See [CTKIP].

Dynamic Symmetric Key Provisioning Protocol (DSKPP) Proposal –DSKPP is a proposed standard for key provisioning that is based on the OATH-developed DSKPP Internet Draft and CTKIP. It is the target IETF standard provisioning protocol that OATH sponsors. See [DSKPP] and [DSKPP1].

Existing Software Provisioning Protocols

This section describes existing methods and protocols for downloading software modules to devices.

Browser based download over HTTPS - Software modules can be downloaded to most devices using a standard mobile Internet browser over HTTPS.

Download Over-The-Air (DLOTA) protocol - The DLOTA protocol is defined by the OMA Forum. It defines a protocol for discovering and downloading content and applications to mobile devices. The protocol can be used to download authentication token software modules. DLOTA security relies fully on security in the transport layer, i.e. it uses HTTP basic authentication. See [DLOTA].

Java MIDP OTA provisioning - The Java MIDP 2.0 download protocol allows discovery and delivery of Java MIDlets to Java devices. See [MIDPOTA].

GSM 03.48 applet download - GSM 03.48 defines a secure protocol for over-the-air delivery and subsequent life-cycle management of SIM applets to SIM cards. See [GSM0348].

6.4.4. OATH Focus Areas

OATH has chosen the following focus areas.

-
- Existing protocols (such as XKMS) can be used to provision public-key information and shared secret keys. However, given the limitations of low-end devices, such as SIM cards and some mobile phones, these protocols may not be suitable. Therefore, OATH will evaluate the requirement for standardizing one or more provisioning protocols for specific credential types.
 - Different credential provisioning protocols have different requirements for handling user, token, and credential identification. OATH will explore the requirement for standardizing a common mechanism for managing these identities.
 - For symmetric credentials, including OTP secrets, OATH has identified requirements and created a draft provisioning protocol [DSKPP1]. OATH will work with the IETF KeyProv working group to standardize a symmetric-key provisioning protocol, where DSKPP is submitted as one of the input documents. A combined protocol from DSKPP1 and CTKIP has been drafted with the same name, DSKPP, under the IETF KeyProv working group. A draft that has features from both proposals and additional enhancements is under review.
 - Different provisioning protocols may have different container formats for a securely protected credential. OATH will explore the requirement for standardizing a common format for symmetric key containers. OATH has submitted a standard format Internet Draft [PSKC] to the IETF KeyProv working group. OATH will work with the IETF KeyProv working group to standardize a portable symmetric-key container, where PSKC is the input document for this effort.

6.5. Common Data Model

The requirement for a common data model derives from two OATH principles.

- Minimize impact on existing infrastructure, and leverage existing infrastructure
- Drive interoperability and best-of-breed solutions

User store schema extensions - Most enterprises have expended significant resources in the last few years to centralize their user stores; both employee directories and customer databases. By standardizing the user store schema extensions required for strong credentials, and working with directory vendors to make this part of their default schema, OATH will minimize the impact of strong authentication on the user directory. Additionally, this will help to standardize the provisioning and management of credential attributes, thereby allowing the enterprise to deploy a best-of-breed solution for the provisioning, and life-cycle management, of credentials that is integrated with the rest of the enterprise infrastructure.

Token metadata - Standardized token metadata will complement the validation and provisioning frameworks described above. Token metadata standardization will enable vendors to ship token information, including key material (e.g. shared secrets) in standardized formats that can be imported into validation and provisioning modules supplied by other vendors.

6.5.1. Existing Standards and Technologies

This section describes existing standard data models.

InetOrgPerson - The inetOrgPerson object class is a general-purpose object class for holding attributes of people. The attributes it holds were chosen to accommodate information requirements found in typical Internet and intranet directory service deployments. The inetOrgPerson object class is designed to be used within directory services based on the LDAP and the X.500 families of protocols, and it should be useful in other contexts as well. It is not mandatory that directory-service implementers use the inetOrgPerson object class; it is simply presented as a well-documented class that implementers can use if they wish. See [RFC2798] and [RFC2251].

PKCS#5 XML – PKCS#5 defines a general key container format that can be used to store token metadata. PKCS#5 XML specifies the PKCS#5 data format in both ASN.1 and XML formats. See [PKCS5XML].

PKCS#12 – PKCS#12 defines a general key-transfer container format that can be used to store token metadata. PKCS#12 specifies the data format in ASN.1. It is mainly used to transfer the private key of an asymmetric key-pair. However, it also specifies a format for a symmetric key. See [PKCS12].

6.5.2. OATH Focus Areas

OATH has chosen the following focus areas.

- OATH plans to encourage development of standardized schema extensions for common user stores such as LDAP directories.
- OATH intends to encourage development of standardized portable token metadata formats that will complement its provisioning and validation frameworks. Specifically, OATH has proposed a standard key container format to the IETF, called Portable Symmetric Key Container (PSKC). The Internet draft specification is an input document to the IETF KeyProv working group, where it will be included as one of the group's deliverables.

6.6. Authentication and Identity Sharing

Most strong authentication solutions today are deployed on an enterprise-by-enterprise basis and used only for a single application. There are several issues with such an approach – each enterprise needs to provision a separate credential to all their users, thereby increasing the cost of the deployment. Additionally, users have to potentially carry multiple devices resulting in poor user experience.

Therefore, if a single authentication token can be used across many sites, then it is much more likely that the consumer will begin to carry it around as a necessary personal tool, much like a cell phone, car keys or credit cards.

Token sharing can occur in several different ways, the two main ways are 'authentication sharing' and 'identity sharing'.

One approach could be to enable sharing of the authentication credential such as an OTP token as an anonymous second factor that can be layered on top of your existing identities and thereby strengthening your various identities. We call this approach as 'authentication sharing'.

In the second approach, identity sharing technologies such as federated identity technologies are used to assert the user's *strong identity* across organizations. We refer to *strong identity* as an identity that is backed by a strong authentication method such as an OTP token or a smartcard.

The liability model for the first approach is much simpler since the only thing that is shared across organizations is a second factor strong authentication token that can be anonymous. The second approach requires a higher level of liability risk for the participating organizations since you are sharing user identities. In the first model each organization is typically responsible for vetting the user's identity, while in the second approach the *relying parties* are relying on the identity vetting performed by the asserting *Identity Provider or IdP*.

While Identity sharing is a mature and well-understood concept, we are introducing the concept of a shared second factor as a lighter-weight alternative to full identity sharing, and it may be a more appropriate solution for strengthening authentication levels across consumer applications.

6.6.1. Authentication Sharing

Sharing second factors is a new approach that enables a network of Web-sites to leverage each other's authentication solutions, and thereby reduce the cost and time needed to deploy consumer strong authentication.

Models

There are several models that can facilitate such a token-sharing 'network' and allow ubiquitous strong authentication for consumers. This section will discuss three different models:

- Centralized Token Service Model
- Distributed Validation Model
- Credential Wallet

The working assumption for these models is that second factor validation can typically be performed at only one place, typically by the token-issuer. However, the models are agnostic in respect of the specific authentication method (one-time-password, certificate, challenge / response, etc.) that provides the second factor.

Centralized Token Service Model

The key concept in the first token-sharing model is a centralized token service infrastructure. See Figure 7. This centralized token service is responsible for provisioning and validating the second factor credential – typically a certificate or an OTP from a hardware token. The token implementing the second factor can be activated by multiple applications or Web-sites. Each application manages a separate first factor – typically a username and password in its own user store. As part of the token activation life-cycle phase, the application stores a mapping between the local username and the common second factor token (e.g. by means of the serial number).

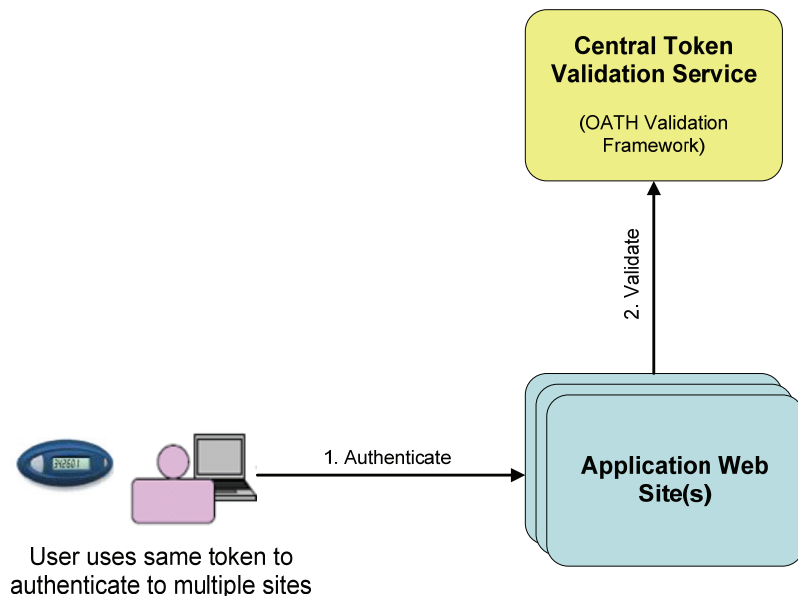


Figure 7 - Centralized token service model

The primary advantage of this model is its ease of deployment. By pushing the infrastructure for second factor support into the service itself, the additional infrastructure that each application needs to deploy is minimal (just a validation proxy to the service). Since the applications are just sharing the anonymous

second factor and not the user identity information, this model entails a relatively simple liability model, which focuses more on the use of the token and its distribution and contents. On the other hand, one of the limitations of this model is that the application will be dependent on the centralized service for second factor validation.

Distributed Validation Model

The second token-sharing model balances the operational need for 'control' with the business need for 'sharing' strong authentication. This model enables token-sharing in a distributed fashion without requiring a centralized token service, as in the first model. It is based on the familiar Domain Name System (DNS).

The key concept of this model is the Token Lookup Service (or TLS) that is analogous to the root DNS server. The token lookup service stores the mapping of the token serial number and the IP address of the validation server that can validate the second factor credential (OTP, etc.) obtained from that token. We will call this validation server the Authoritative Validation Node (AVN) for that token serial number. This model is shown in Figure 8.

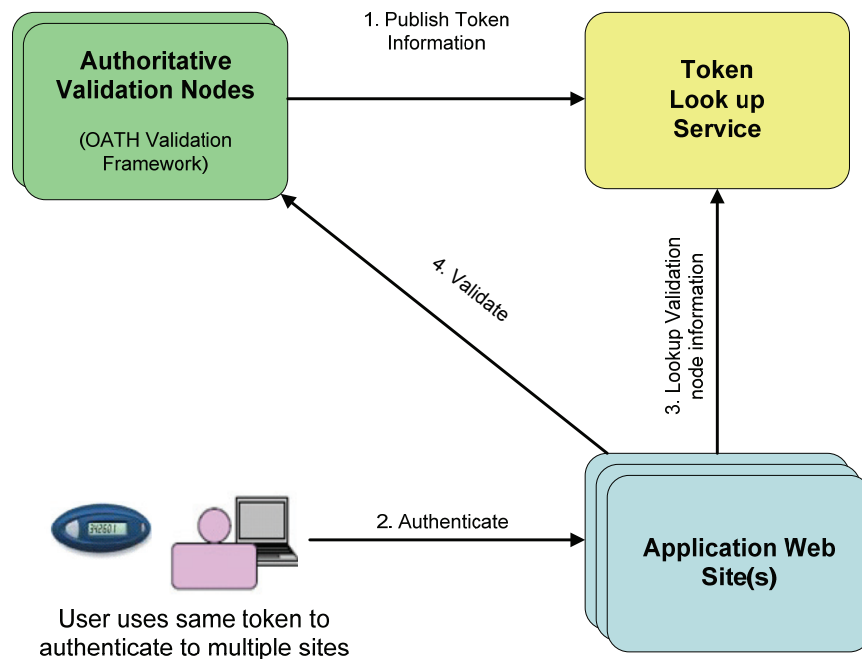


Figure 8 - Distributed validation model

This model has been inspired by DNS and hence it can leverage proven technology and business and deployment models. One of the primary advantages of this model is that it enables an organization to implement strong authentication in isolation, joining a broader authentication network at some point in the future. From the operational point of view, in this model, the application Web-site becomes dependent on a third-party authentication infrastructure for its second-factor validation. Additionally, different token-issuing parties may have deployed infrastructures with varying levels of sophistication.

Credential Wallet

This model leverages next-generation devices, like Java cell phones, PDAs and USB flash drives that have storage and application capabilities and some form of built-in or external graphical interface for managing credentials. See Figure 9. In this model, the device becomes a 'wallet' that can contain

multiple instances of a second-factor credential. For each site that needs strong authentication, the necessary credential can be selected.

Each site can provision, manage and validate its own instance of the second factor credential, which is typically an OTP token, on the same physical device; thereby obviating the need for users to carry multiple tokens.

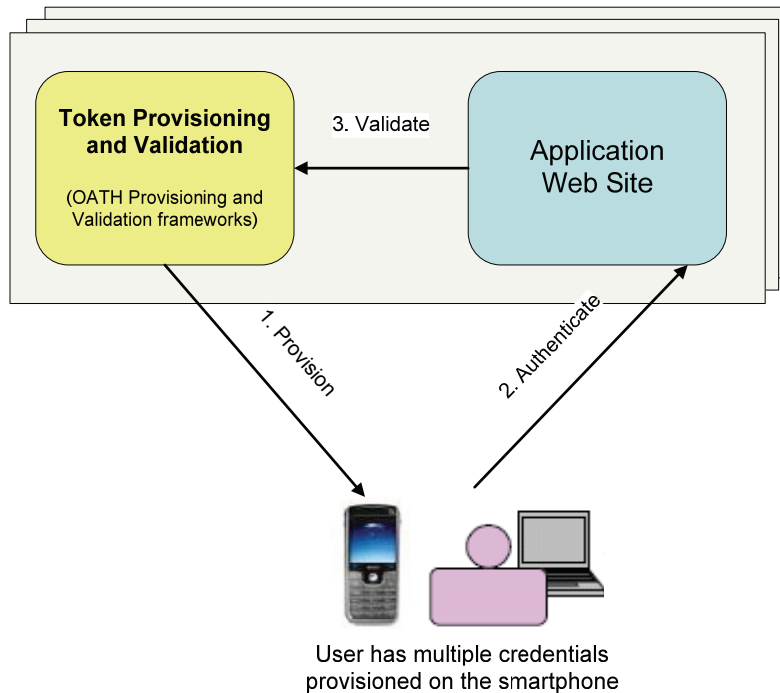


Figure 9 - Credential wallet

The primary advantage of this model is that the second-factor tokens are not shared, and hence, each organization has the flexibility to deploy and manage its own second-factor authentication solution. There are no external dependencies for validating the second-factor credential. The support and operational models are relatively simple. However, in this model, each organization has to issue, deploy and manage second-factor credentials for its users. Another limitation of this model is that, at the current time, not all users have next-generation devices.

OATH Focus Areas

- Common token identifier namespace
- Protocol for token lookup service
- Provisioning protocols that enable provisioning of credentials to devices such as cell phones, USB drives, etc.
- A Web-services validation protocol specification that enables validation of a variety of authentication methods across networks.

6.6.2. Identity Sharing

Sharing of identity comprises all standards and technologies that enable the portability of identity information across otherwise autonomous security domains. The identity provider (or IdP) is the domain that authenticates the user identity and provides identity assertions to other domains (relying parties). The identity assertion typically contains one or more attributes of the user identity.

This approach effectively enables the sharing of second-factor credentials across domain boundaries, provided that the user is required to authenticate with a second-factor token to the identity provider. The identity assertion typically includes an 'authentication context'. The authentication context is a set of attributes that describes the credentials that the IdP used to authenticate the user in the first place. For example, the IdP may add an authentication context to indicate that the user was authenticated using a username / password and a one-time-password token.

Figure 10 shows a typical federated identity scenario.

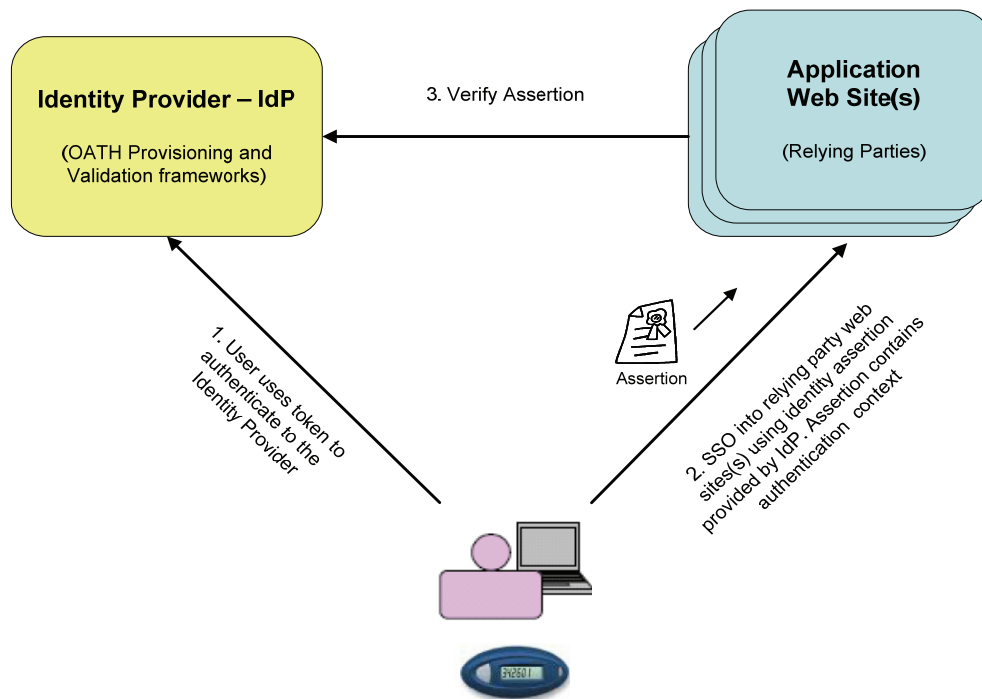


Figure 10 - Federated identity

There are several types of identity-sharing – the traditional 'enterprise-controlled' or B2B approaches, such as Liberty, as well as the more recent 'user-controlled' or 'user-centric' approaches such as OpenID and CardSpace.

6.6.3. Traditional Federated Identity

Traditional approaches to identity-sharing on the Internet have been formalized through technologies such as [SAML] and Liberty Alliance [LBTY] standards. These technologies enable use cases such as cross-domain Web-based single sign-on; cross-domain user-account provisioning, cross-domain entitlement management and cross-domain user attribute exchange. In these scenarios, the Identity Provider typically controls with whom, where and when the identity may be shared. This has, typically, worked well in enterprise-controlled, or B2B, scenarios.

6.6.4. User-centric Identity Sharing

More recently, another approach to identity-sharing has emerged – called user-centric identity or Identity 2.0. The major difference between this and the traditional approach is, typically, that there is an identity agent that participates in every transaction on behalf of the user. The identity agent may be implemented on the client (as in Microsoft’s CardSpace) or it may be hosted remotely (as in OpenID). There are three major advantages to this approach:

- The user has absolute control over how, where, when and what identity attributes are shared.
- Scalability - there is no need for relying parties to be pre-registered with the identity provider.
- Usability – the user has a consistent user experience.

Figure 11 shows the CardSpace Identity Selector UI that lets the user choose which identity to disclose to the relying party. Here, the user’s identity agent is manifested as a client on the user’s computer.

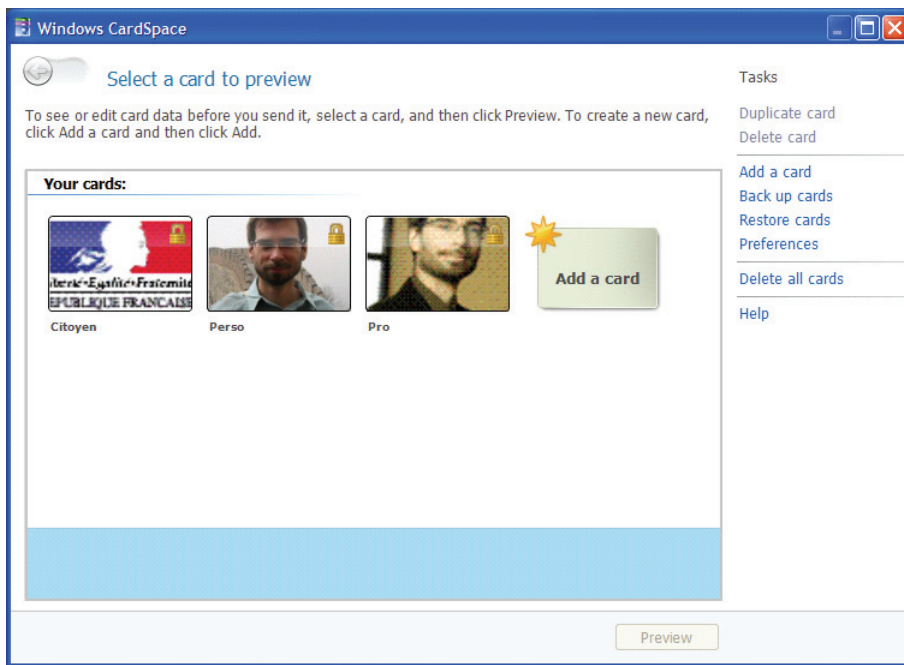



Figure 11 - CardSpace identity-selector user interface

Figure 12 shows the user interaction for OpenID. Here, the user’s identity agent is manifested as a remote URL.

Sign In Using Your OpenID

OpenID URL:

 OpenID lets you safely sign in to different websites with a single password. [Get an OpenID.](#)

SIGN IN WITH OPENID

Figure 12 - OpenID user interface

6.6.5. OATH Focus Areas

There are several industry initiatives that enable identity-sharing or federated identity. Hence, OATH will focus on collaborating, and making sure that OATH credentials are supported as “first-class citizens” in emerging standards and technologies, such as CardSpace and OpenID.

Additionally, OATH will look at standardizing some attributes for use in the authentication context component of an identity assertion. This will ensure that all applications and relying parties can expect a consistent experience when the user uses an OATH credential to authenticate to the Identity Provider.

7. Example Deployment Scenario

In this section, we provide an example to illustrate how an implementation of the OATH reference architecture might be deployed in the real world. In this example, a fictitious bank, called MyBank, has deployed two-factor authentication for its retail banking customers. The bank has a mixed user population, with three user categories.

- The first category contains mobile users who need to access their banking application from multiple locations. Also, this set of users has next-generation mobile phones that are capable of acting as a container for OTP credentials.
- The second category contains users who mainly access their bank from a single location.
- The third category contains users who already have tokens from Vendor A, provisioned as part of a corporate banking application.

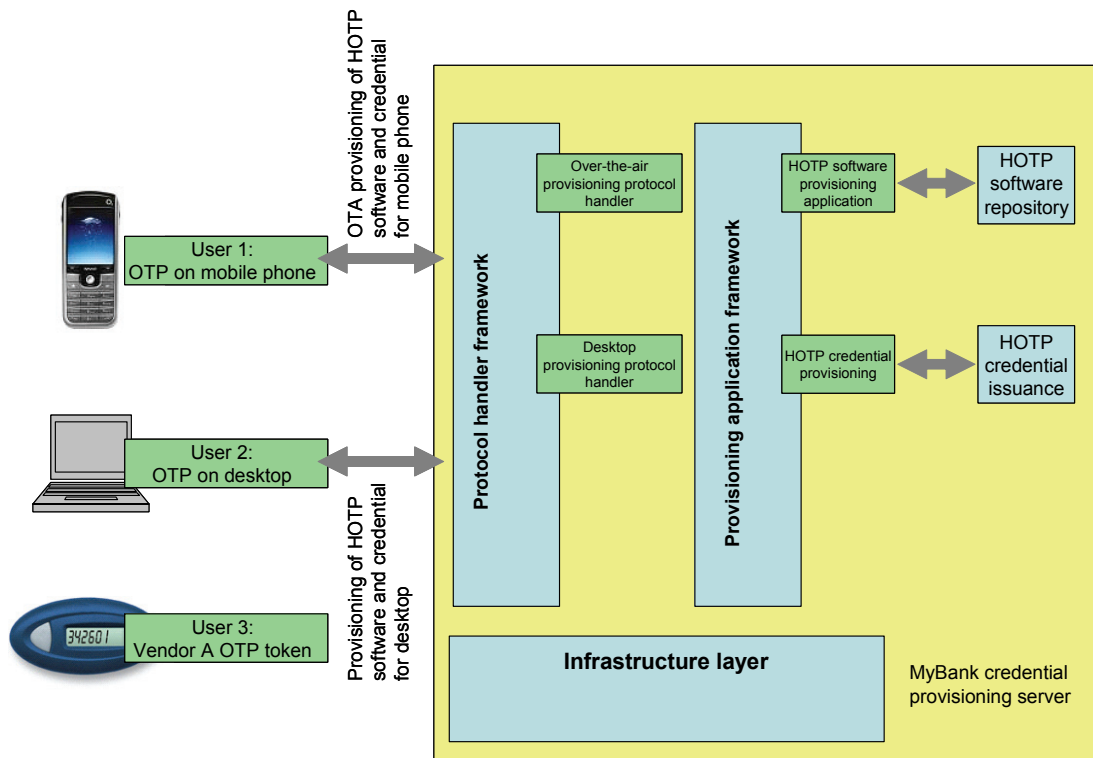


Figure 13 - MyBank provisioning infrastructure

Figure 13 shows the provisioning infrastructure deployed by MyBank. MyBank has deployed two provisioning protocol handlers. One can provision authentication software and credentials over-the-air to mobile devices. The second handler can provision authentication software and credentials to desktop PCs. MyBank has also deployed two provisioning applications: a HOTP software provisioning application, and a HOTP credential provisioning application. These applications talk to the HOTP software repository and the HOTP credential issuer, respectively.

As shown in the figure, there are three users, representing each user category. User 1 has a mobile phone, and is provisioned with the necessary HOTP software token and HOTP credential (shared secret) using the OTA provisioning protocol. User 2 is similarly provisioned with the necessary HOTP software token for the desktop and the corresponding HOTP credential, using the desktop provisioning protocol.

Finally, User 3 already has an OTP token from Vendor A. At this point, all three representative users can access MyBank's retail banking application using strong authentication.

Figure 14 shows the validation infrastructure deployed by MyBank. The validation deployment has two different protocol handlers that can accept OATH-WebServicesValidationProtocol and RADIUS validation requests. Also, MyBank has validation handlers to validate the two different flavors of OTP credentials (HOTP and Vendor A's OTP tokens) that have been deployed to the user population. It also has a username / password validation handler that the application can leverage.

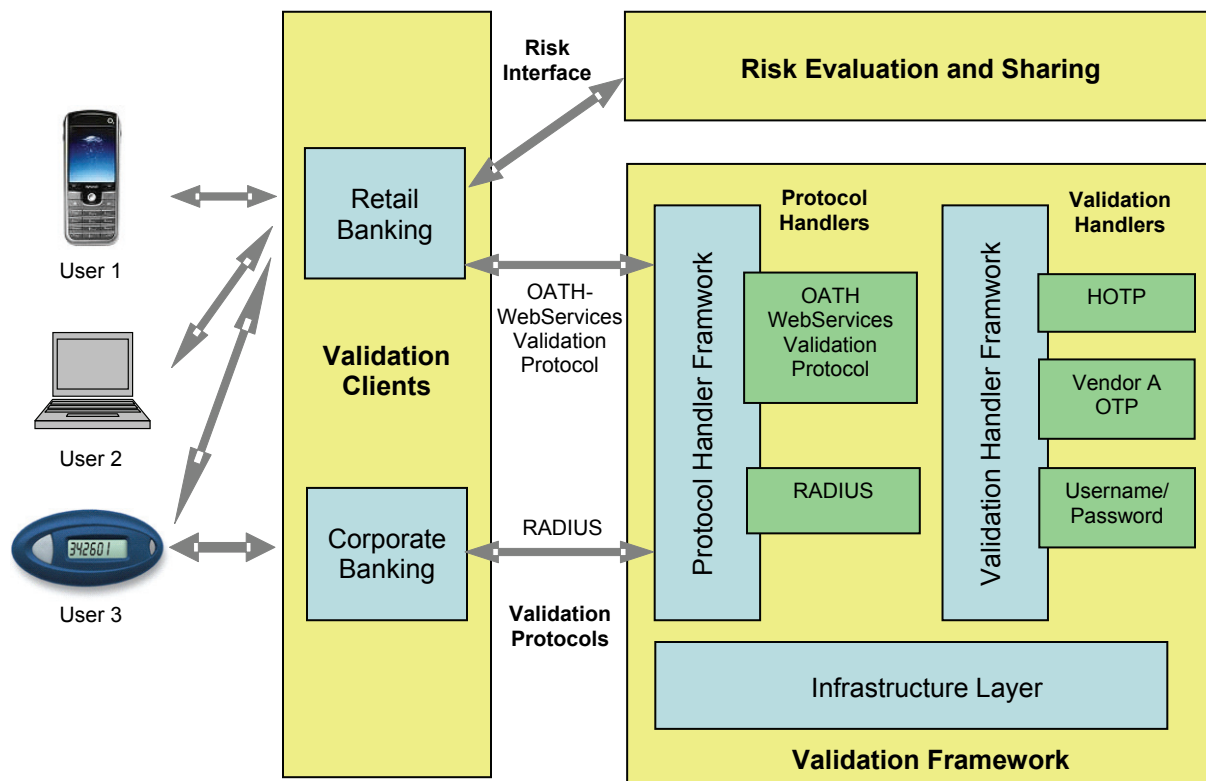


Figure 14 - MyBank validation infrastructure

There are two applications that make use of this risk and validation infrastructure: a retail banking Web application that communicates with the validation server using the OATH-WebServicesValidationProtocol, and a corporate banking Web application that uses RADIUS as its validation protocol.

Additionally, the retail banking application also uses a risk-based approach to authentication by using the *risk evaluation and sharing* module. In this approach the user is required to use an OTP credential when the risk is perceived to be high. On the other hand the corporate banking web application requires the use of an OTP credential at all times.

When the user connects with the retail banking web application the request is first passed to the *risk evaluation and sharing* module using the risk interface. If the risk is perceived to be low, then the user may log in to the application using username and password only. On the other hand if the request is deemed to be high risk (e.g. from an unknown terminal) then the user is requested to enter an OTP in addition to the username and password.

As shown in the figure, all three representative users can use the OTP values generated by their respective tokens (viz. HOTP soft token on the mobile phone, HOTP soft token on the desktop and the

standalone token from Vendor A). The validation server will route the request to the appropriate handler to validate the user credentials.

Note that the HOTP software token may have multi-key capabilities. In this case, the user can be provisioned with more than one instance of the HOTP credential; one from MyBank and the other one from a brokerage service or ISP, turning the user's mobile phone or desktop into a credential wallet.

User 3 can, additionally, continue to log into the corporate banking application using Vendor A's OTP token. The validation request will be received by the RADIUS protocol handler and routed to the Vendor A validation handler to be validated.

As this example shows, the OATH reference architecture enables MyBank to consolidate its various credential provisioning, management and validation systems into a single infrastructure that can service its diverse user population and application requirements.

8. Summary of OATH Focus Areas

The reference architecture described above, addresses five main architectural components: client framework, validation framework, risk-evaluation framework, provisioning / management framework and common data model.

Focus areas that OATH has identified are summarized below.

Client Framework

- **Authentication methods** - OATH will encourage the standardization of an OTP algorithm to enable client / server interoperability for two-factor authentication, with extensions for event-based, time-based and challenge / response variants such as OCRA. Additionally, OATH will promote standard, low-cost authentication methods for consumer usage.
- **Authentication tokens** - OATH will foster innovation in tokens by embedding authentication technologies into devices that users carry for other purposes, such as mobile phones, and by providing flexibility and cost savings through multi-function tokens and multi-key tokens. OATH will also consider the requirement for standards governing other aspects, such as a namespace for the token identifier, based on the IEEE EUI-64 standard, in order to improve interoperability among vendors.
- **Token interface** - OATH will promote extensions to existing APIs to support OTP algorithms, such as HOTP, and define new token interface APIs, including a standard software token OTP API, and authentication-specific HTML tags for Web applications.
- **Authentication protocols** - OATH is currently investigating the creation of a Web-services-based standard authentication protocol. OATH's aim is to promote, either by extension or creation, a royalty-free protocol that supports validation of authentication requests with a variety of methods, as described above. OATH will research the use of sideband signaling through separate IP channels (e.g., wired and wireless) to address more advanced account hijacking threats associated with ID theft and phishing / pharming. It will define HTML tags to identify authentication-related data elements in Web applications.

Validation Framework

- **Pluggable validation handlers** - OATH will promote the development of appropriate interfaces (for both protocol and validation handlers) that will enable vendors to write pluggable validation handlers. In particular, it will define JAAS Modules for HOTP and OCRA.
- **Web-services validation protocol** - OATH will develop a Web-services validation protocol specification suitable for a variety of authentication methods.
- **Credential-specific validation protocols** - OATH will evaluate the requirement for either specifying standardized extensions to existing validation protocols (e.g. RADIUS) or standardizing one or more additional validation protocols that target specific credential types.

Risk evaluation framework

- **Fraud-pattern exchange** - OATH will complete development of a Web-services interface specification for exchanging identity-fraud pattern data.
- **Risk interface** - The risk interface has to be defined. It allows the validation framework (or application) to pass relevant information about the request/transaction and obtain, in return, a risk score and/or identifiers for the set of authentication methods that both satisfy the risk requirement and are available to the user.

-
- **Validation protocol extension point** - Extension points must be added to both the validation protocol (OATH-WebServicesValidationProtocol) and the risk interface to allow the application to pass the necessary context information to the risk component (schema for the context information will not be defined at this stage).
 - **Language bindings** - Bindings for both Java and Web-services are required for the risk interface.
 - **Database schema** - The need to define database schema to contain the identifiers for the authentication methods available to the user will be explored.

Client Provisioning and Management Framework

- **Framework for existing provisioning protocols** - OATH will promote the development of a framework for existing standard-based provisioning protocols that will enable vendors and customers to deploy proprietary provisioning protocols for specific types of credentials and devices.
- **Standardized provisioning protocols** - OATH will participate in the development of a standard key provisioning protocol as part of the IETF KeyProv working group formed at the start of 2007.

Common Data Model

- **User-store extensions** - OATH will promote the definition of standard user-store extensions such as LDAP directories.
- **Token metadata** – OATH will propose standard formats for OTP token metadata to support open authentication, including the profiling of EUI-64 as a token namespace identifier. It will complete the standardization of PSKC in the context of the IETF KeyProv working group.

9. References

- [CAP04] MasterCard International Incorporated, "Chip Authentication Program - Functional Architecture", September 2004
- [CKM_HOTP] "PKCS#11 Mechanisms for One-Time Password Tokens", available at: <http://www.rsasecurity.com/rsalabs/node.asp?id=2818>
- [CTKIP] "Cryptographic Token Key Initialization Protocol", draft 2, 14 April 2005, RSA Laboratories, available at: <ftp://ftp.rsasecurity.com/pub/otps/ctkip/ctkipv10d2.pdf>
- [DHEKE] W. Diffie, M.E. Hellman, "New directions in cryptography", IEEE Trans. Inform. Theory, IT-22, 6, 1976, pp.644-654
- [DLTA] "Generic Content Download Over The Air", approved version 1.0, 25 June 2005, Open Mobile Alliance, available at: http://www.openmobilealliance.org/release_program/download_v10.html
- [DSKPP] M. Nystrom, S. Machani, M. Pei, A. Doherty, "Dynamic Symmetric Key Provisioning Protocol", IETF Internet Draft, available at: <http://tools.ietf.org/id/draft-doherty-keyprov-dskpp-00.txt>
- [DSKPP1] M. Pei, S. Machani, "Dynamic Symmetric Key Provisioning Protocol", IETF Internet Draft, available at: <http://tools.ietf.org/id/draft-pei-keyprov-dynamic-symkey-prov-protocol-00.txt>
- [EUI64] "Guidelines for 64-bit global identifier (eui-64) registration authority", IEEE, available at: <http://standards.ieee.org/regauth/oui/tutorials/EUI64.html>
- [GSM0348] "Security Mechanisms for the SIM application toolkit", GSM 03.48, version 8.2.0 Release 1999, Digital cellular telecommunications system (Phase 2+)
- [HOTP] D. M'Raihi et al, "HOTP: An HMAC-based One Time Password Algorithm", available at: <http://www.ietf.org/internetdrafts/draftmraihihoathhmacotp03.txt>
- [JAAS] "Java Authentication and Authorization Service", available at: <http://java.sun.com/products/jaas/>
- [LBTY] "The Liberty Alliance", available at: <http://www.projectliberty.org/>
- [MIDPOTA] "Mobile Information Device Profile 2.0", JSR 118, Java Community Process, available at: <http://jcp.org/aboutJava/communityprocess/final/jsr118/>
- [MINA] "Multipurpose Infrastructure for Network Applications", Apache Directory Project, available at: <http://directory.apache.org/subprojects/network/index.html>
- [MSCNG] "Cryptographic API: Next Generation", Microsoft Corporation, available at: <http://msdn2.microsoft.com/en-us/library/aa376214.aspx>
- [OPNI] "What is OpenID?" Available at: <http://openid.net/>
- [MSFT] "What is Windows CardSpace?" Available at: <http://cardspace.netfx3.com/>
- [PAM] "Pluggable Authentication Module", available at: http://www.freebsd.org/doc/en_US.ISO8859-1/articles/pam/

[PKCS5] “Password-Based Cryptography standard”, RSA Laboratories, available at:
ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-5v2/pkcs5v2_1.pdf

[PKCS5XML] “Amendment 1: XML Schema for Password-Based Cryptography”, RSA Laboratories, available at: <ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-5v2/pkcs-5v2-0a1.pdf>

[PKCS11] “Cryptographic Token Interface Standard”, RSA Laboratories, available at:
<ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-11/v2-20/pkcs-11v2-20a3.pdf>

[PKCS12] “Personal Information Exchange Syntax Standard”, RSA Laboratories, available at:
<ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-12/pkcs-12-tc1.pdf>

[PSKC] P. Hoyer, M. Pei, S. Machani, A. Vassilev, J. Martinsson, “Portable Symmetric Key Container”, IETF Internet Draft, available at: <http://tools.ietf.org/id/draft-hoyer-keyprov-portable-symmetric-key-container-00.txt>

[RFC2315] B. Kaliski, “Cryptographic Message Syntax Version 1.5”, IETF, RFC 2315, March 1998, available at: <http://www.ietf.org/rfc/rfc2315.txt>

[RFC2510] C. Adams, S. Farrell, “Internet X.509 Public Key Certificate Management Protocols”, IETF, RFC 2510, March 1999, available at: <http://www.ietf.org/rfc/rfc2510.txt>

[RFC2511] M. Myers, C. Adams, D. Solo, D. Kemp, “Internet X.509 Certificate Request Message Format”, IETF, RFC 2511, March 1999, available at: <http://www.ietf.org/rfc/rfc2511.txt>

[RFC2616] “Hypertext Transfer Protocol – HTTP/1.1”, IETF, RFC 2616, June 1999, available at:
<http://www.ietf.org/rfc/rfc2616.txt>

[RFC2617] “HTTP Authentication”, IETF, RFC 2617, June 1999, available at:
<http://www.ietf.org/rfc/rfc2617.txt>

[RFC2630] R. Housley, “Cryptographic Message Syntax”, IETF, RFC 2630, available at:
<http://www.ietf.org/rfcs/rfc2630.html>

[RFC2797] M. Myers, X. Liu, J. Schaad, J. Weinstein, “Certificate Management Messages over CMS”, IETF, RFC 2797, April 2000, available at: <http://www.ietf.org/rfc/rfc2797.txt>

[RFC2986] M. Nystrom, B. Kaliski, “PKCS #10: Certification Request Syntax Specification, version 1.7”, IETF, RFC 2986, November 2000, available at: <http://www.ietf.org/rfcs/rfc2986.html>

[SAML] “Assertions and Protocols for the OASIS Security Assertion Markup Language”, v2.0, OASIS Standard, 15 March 2005, available at: <http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf>

[SCEP] Internet Draft, X. Liu, C. Madson, D. McGrew, A. Nourse, “Simple Certificate Enrollment Protocol”, revised 11 Feb 2005

[SPEKE] D. Jablon, “Simple Password Exponential Key Exchange”, September 1996, available at:
<http://www.jablon.org/jab96.pdf>

[THRD] “How to Share Transaction Fraud (Thraud) Report Data”, M'Raihi et al, March 2007, available at: <http://www.ietf.org/internet-drafts/draft-mraihi-inch-thraud-02.txt>

[TPM] “Trusted Platform Module”, Trusted Computing Group, available at:
<https://www.trustedcomputinggroup.org/downloads/specifications/>

[VRSN] “Consumer strong authentication, addressing deployment obstacles by enabling token sharing”, VeriSign, Inc., available at: <http://www.verisign.com/static/029726.pdf>

[WSS] “Web Services Security: SOAP Message Security 1.1”, OASIS Standard Specification, February 2006, available at: <http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf>

[XKMS] “XML Key Management Specification”, 30 March 2001, W3C, available at:
<http://www.w3.org/TR/xkms/>

10. Contributing members

This paper is a result of contributions from members of the OATH Technical Focus Group. The following people were members of the OATH Technical Focus Group at the time of publication.

Amol U. Deshmukh	Gemalto, Corp.
Anthony Nadalin	IBM, Corp.
C Russell	Swivel Secure
Dave Jevans	Ironkey
David Berman	VeriSign, Inc.
David M'Raihi	VeriSign, Inc.
Didier Mobetie	Ncryptone
Don Malloy	Innovative Card Technologies
Dragoljub Nestic	Thales e-Security
Eric Plet	Gemalto, Corp.
Eric Vila	ActivIdentity, Inc.
Ernesto Frutos	Authenex
Fred McClain	Boojum Mobile, Inc.
Frederic Engel	Livo Technologies
Gary Chew	Gemalto, Corp.
Gireesh Subramanya	FNF
Graham Newton	Thales e-Security
Hagai Bar-El	Discretix, Inc.
Henri Quiniou	Citrix Systems, Inc.
J Hamaguchi	Konica Minolta
Jean-Philippe Authier	Ncryptone
Jeff Bohren	BMC Software, Inc.
Jim Spring	Ironkey
Johan Rydell	Portwise
John Stewart	Signify
Jonathan Tuliani	Cryptomathic
Julian Lovelock	ActivIdentity, Inc.
Kevin Lewis	Sandisk
Larry Hamid	MXI Security
Liam Crilly	Signify
Mingliang Pei	VeriSign, Inc.
Ohad Ranen	Aladdin Knowledge Systems
Ophir Shalitin	Discretix, Inc.
Philip Hoyer	ActivIdentity, Inc.
Philippe Buschini	Ncryptone
Phillip Hallam-Baker	VeriSign, Inc.
Rami Elron	BMC Software, Inc.
Rich Skibo	Spyrus
Robert Johnston	Executive Events
Ron Ensh	Citala
Ron LaPedis	Sandisk
Salah Machani	Diversinet Corp.
Sharon Boeyen	Entrust, Inc.
Shiddalinganagouda Rati	HP
Shuh Chang	Gemalto, Corp.
Siddharth Bajaj	VeriSign, Inc.
Steve Anderson	BMC Software, Inc.
Steve Ryan	Ironkey
Stuart Vaeth	Diversinet Corp.
Sudhakar Avula	Symwave

Thomas Fleury
Thomas Le Ouedec
Tim Moses
Todd Inskip
Younghwan Kim

Xelios
Ncryptone
Entrust, Inc.
Bank of America
Cluem