Welcome

To Advance through Presentation
Use Page Up and Page Down Keys

Keychain Manager

Ken McLeod
Senior Engineer

Overview

- What is the Keychain and how does it work?
- What can you do with it?
- Obtaining a code-signing certificate
- Using the Keychain Manager API
- Demo
- Q & A

What Is The Keychain?

- Secure repository for passwords, cryptographic keys, and digital certificates
  - Multiple data stores
  - Currently file-based
- Built on industry-standard CDSA framework
- Provides transparent authentication to services (single sign-on)

Keychain Terminology

- A Keychain is either locked or unlocked
    - Data can only be retrieved from an unlocked Keychain
- Each stored password, key, or certificate is a Keychain item
- Default Keychain is the one to which new Keychain items are added

Keychain Usage

- A system can have multiple Keychains (typically one per user)
- More than one Keychain may be unlocked at a time
- Keychains can be set to lock automatically

Keychain Usage (Cont.)

- Individual Keychain files are portable to other systems
- Keychains can be located on AFP servers and locked media
- Will support smart cards and removable tokens in the future

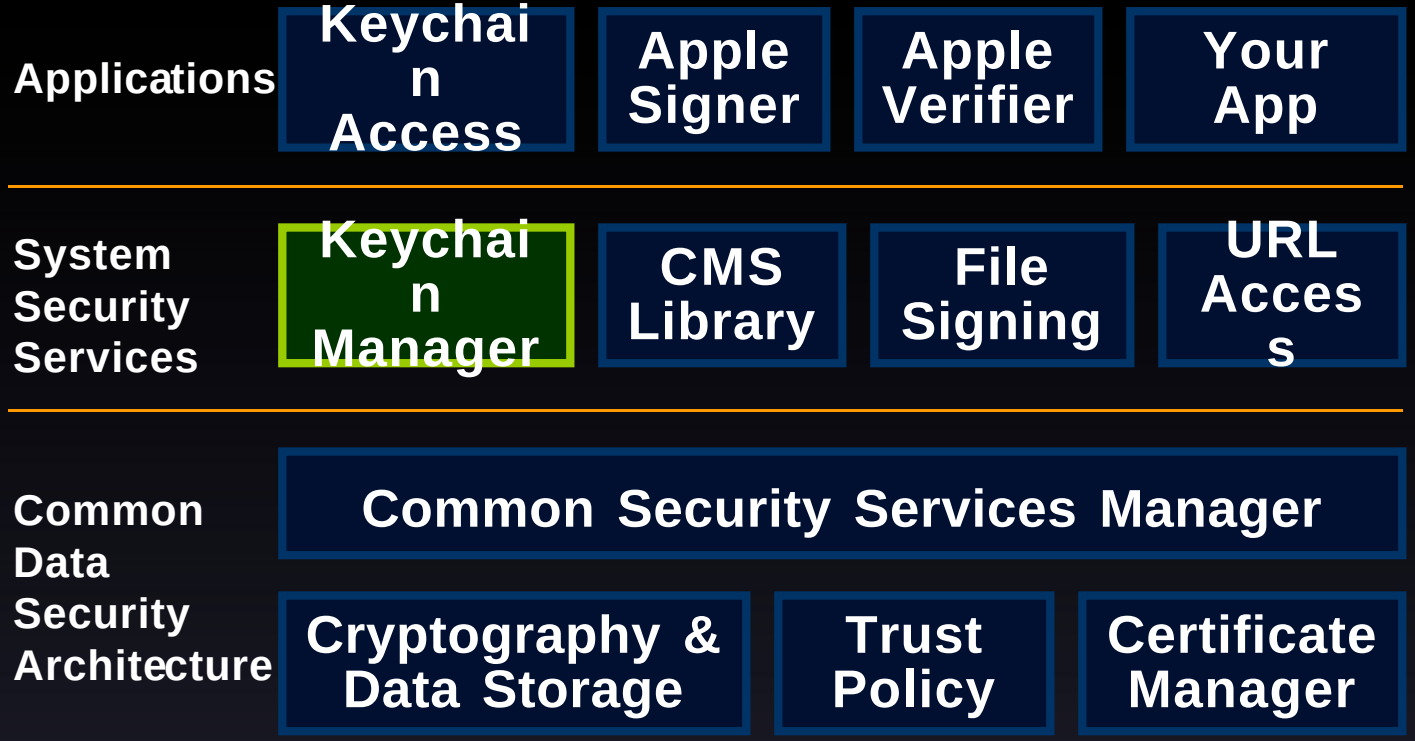
Technical Details

- PowerPC machines only
- Can be called from 68K or PowerPC code—Pascal, C, or C++
- Fully scriptable API
- Requires Appearance Manager, Drag Manager, Navigation Services, Core Foundation Library

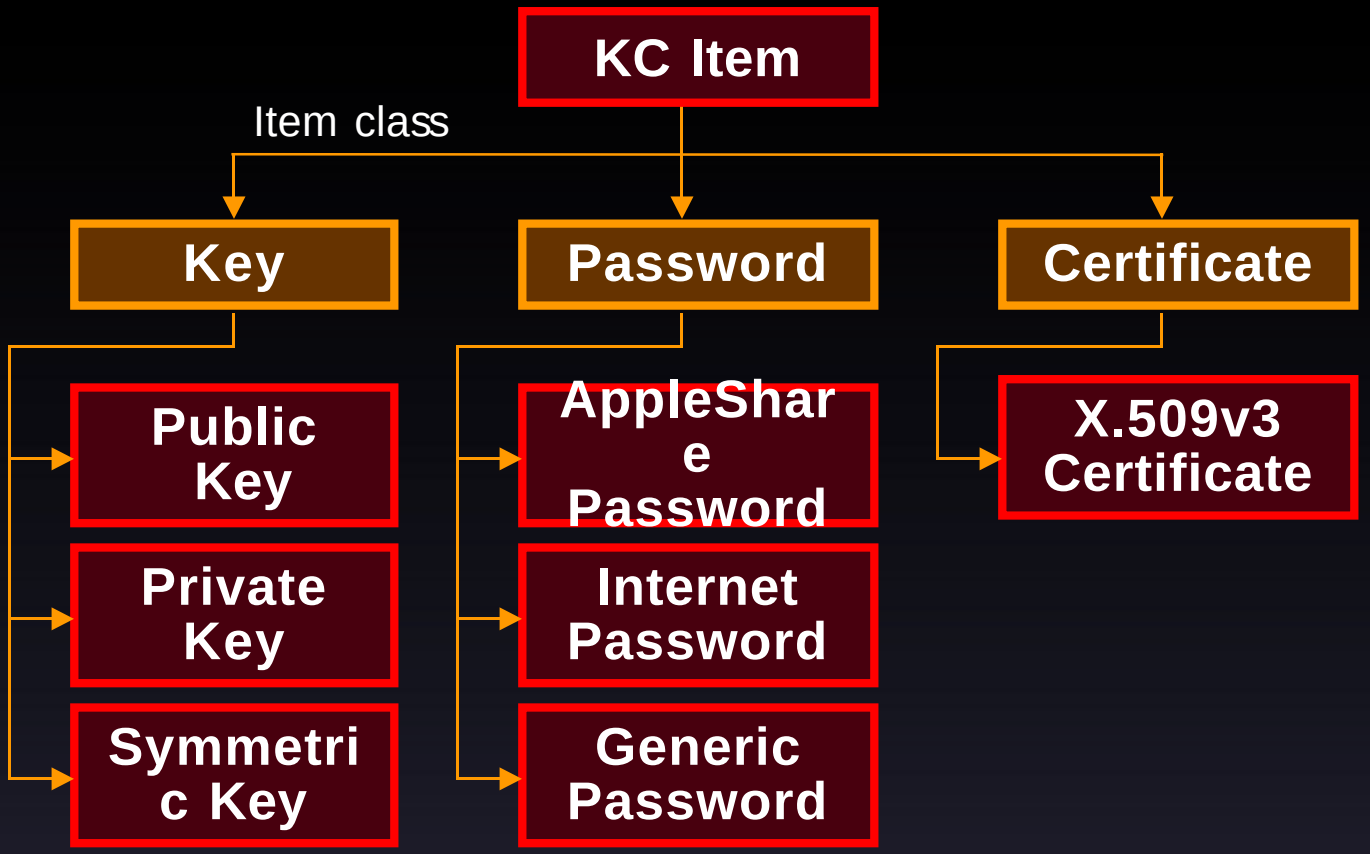
Where We're At

- Now part of the Mac OS!
- Backward-compatible with Keychain 1.0 SDK (with the exception of KCCreateKeychain)

Keychain and CDSA



Keychain Items



What's In an Item?

- Two components: attributes and data

Account	"ken"
Server	"ftp.apple.com"
Path	"/private/stuff/"
Protocol	ftp

"thinkdifferent"

Item Attributes

- All items have common attributes
  - Item class (e.g. “Internet password”)
  - Date created and modified
  - Label, description, and comment
- Each item class has additional attributes
- Attributes are used to find an item
- Attributes are encrypted

Item Data

- Cannot be searched or used to retrieve an item
- Cannot be retrieved unless the Keychain is unlocked
- Cannot be retrieved without the user's explicit permission
- Private and symmetric key data cannot be retrieved

Internet Password

- Attributes
  - Account name (“ken”)
  - Server name (“ftp.apple.com”)
  - Path (“/private/stuff/”)
  - Protocol, Port , Security Domain
- Data
  - Password string

AppleShare Password

- Attributes
  - Account Name (“Ken”)
  - Server Name (“Ken’s Machine”)
  - Server Address (zone name, domain, or IP address)
  - Volume (optional)
- Data
  - AFPXVolMountInfo structure

Certificate Item

- Attributes
  - Certificate type
  - Subject (name, e-mail address)
  - Issuer
  - Serial Number
  - Date issued and expiration date
- Data
  - X.509 certificate (BER-encoded)

Key Item

- Attributes
  - Algorithm (e.g. RSA, DSA, FEE)
  - Key size (e.g. 128-bit)
  - Key usage (sign, encrypt, wrap)
  - Date issued and expiration date
- Data
  - Public keys: the key value
  - Private/symmetric keys: not available!

How Secure Is It?

- Much better than status quo
  - No need to write passwords down or store them insecurely on disk
- Keychain password is never stored on disk
  - Symmetric encryption key is derived from the Keychain password
- Export-approved 128-bit RC2 encryption for “access control” storage

How Secure Is It? (Cont.)

- Keychain is normally unlocked with user interaction
- Programmatic unlock is allowed until first failure (once per restart)
- Delay between failed authentication attempts grows exponentially
- Prevents systematic attacks

How Secure Is It? (Cont.)

- Private Keychain items cannot be retrieved without the user's explicit permission
  - Permission can be given on a per-item, per-process, or per-Keychain basis
  - Prevents rogue applications from stealing passwords while Keychain is unlocked
- Keys optionally require a usage password

What Can You Do with It?

- Store your application's passwords securely in a Keychain instead of a preferences file
- Applications can share passwords for common services
- Store keys and digital certificates for use with code signing, secure e-mail and other applications

Obtaining a Certificate

- Go to a certificate authority's Web site (Thawte, VeriSign) using Navigator
- Navigator generates a key pair and sends the public component to the CA
- Export the certificate and key pair from Navigator as a PKCS#12 file
- Import the file to your Keychain

Obtaining a Developer Certificate

Mark Shuttleworth
President, Thawte

Keychain Manager APIs

Craig Mortensen
Senior Engineer

API Overview

- Getting started with high-level calls
- Low-level “building block” routines
- Searching for items
- Managing Keychains
- Notification

Getting Started

- Call **KeychainManagerAvailable**
- Don’t need to explicitly create or unlock a Keychain; let the user do it!
- Find password or certificate using high-level calls

Finding an Item

- Use high-level calls to find a single password or certificate that matches a given set of attributes
  - **KCFindAppleSharePassword**
  - **KCFindInternetPassword**
  - **KCFindGenericPassword**
  - **KCFindX509Certificate**

Adding an Item

- Add a single password or certificate to the default Keychain using high-level calls
  - **KCAddAppleSharePassword**
  - **KCAddInternetPassword**
  - **KCAddGenericPassword**
  - **KCAddX509Certificate**

What High-Level Calls Do

- High-level routines (Add, Find) are implemented using low-level “building block” calls

KCAddInternetPassword =

KCNewItem
KCAddItem
KCSetAttribute
KCSetAttribute
KCSetAttribute
KCSetAttribute
KCSetAttribute
KCSetData

Keychain Items

- Items are created by **KCNewItem**
- Items are added to the Keychain by **KCAddItem**
- Items are accessed through a **KCItemRef**
- Manipulate item attributes using **KCGetAttribute, KCSetAttribute**
- Manipulate item data using **KCGetData,KCSetData**

Keychain Items (Cont.)

- Commit changes to an item using **KCUpdateItem**
- Release the item using **KCReleaseItem** when finished

Searching For Items

- Use **KCFindFirst** only when you expect or want to find more than one item
- Continue searching with **KCFindNext**
- Release the search criteria when finished by calling **KCReleaseSearch**

Managing Keychains

- **KCRef** identifies a Keychain by reference
  - Currently, Keychains are file-based
  - In the future, smart cards and removable tokens will be supported
- **KCCreateKeychain**
- **KCGetStatus**
- **KCCountKeychains**
- **KCGetIndKeychain**

Notification

- Keychain events
    - Unlock, Lock, Add, Find, GetData, Delete, Update, DefaultChanged
- If you want to be notified of Keychain events, install a callback with **KCAddCallback**
- Register only for events you care about
- Use **KCRemoveCallback** when done

Demo

Keychain Does It For You

- Keychain is a secure repository for passwords, keys, and digital certificates, provided for you as a system service
- Easy to use; only a few calls are needed for most applications
- In conjunction with other Security APIs, allows developers to provide secure and signed data

For More Information:

Security APIs: Keychain	Hall C **Wed., 2:30pm**
Security APIs: CMS, and URL Access	Hall C **Wed., 4:00pm**
Data Security Feedback Forums	Hall J2 **Fri., 4:00pm**
SDK Information:	http://www.apple.com/developer/

Q & A

Think different.™

Welcome

To Advance through Presentation
Use Page Up and Page Down Keys