Core OS

#WWDC14

Keychain and Authentication with Touch ID

Learn how you can integrate Touch ID

Session 711

Wade Benson
Core OS Security Engineering

Libor Sykora
Core OS Security Engineering

© 2014 Apple Inc. All rights reserved. Redistribution or public display not permitted without written permission from Apple.

What You’ll Learn

What You’ll Learn

Store user secrets securely (passwords, keys, …)

What You’ll Learn

Store user secrets securely (passwords, keys, …)

New Keychain Data Protection Class

What You’ll Learn

Store user secrets securely (passwords, keys, …)

New Keychain Data Protection Class

New Access Control List (ACLs)

What You’ll Learn

Store user secrets securely (passwords, keys, …)

New Keychain Data Protection Class

New Access Control List (ACLs)

Touch ID

Keychain Features

Keychain Features

A very specialized database

Keychain Features

A very specialized database

Protected data (payload)

Keychain Features

A very specialized database

Protected data (payload)

Keychain attributes

Keychain Features

A very specialized database

Protected data (payload)

Keychain attributes

Efficient search by attributes

Keychain Features

A very specialized database

Protected data (payload)

Keychain attributes

Efficient search by attributes

Optimized for user secrets

Why the Keychain?

Why the Keychain?

Protected with the user passcode

Why the Keychain?

Protected with the user passcode

Protected with the device secret

Why the Keychain?

Protected with the user passcode

Protected with the device secret

Protect secrets at rest

Why the Keychain?

Protected with the user passcode

Protected with the device secret

Protect secrets at rest

Encrypted backup

Why the Keychain?

Protected with the user passcode

Protected with the device secret

Protect secrets at rest

Encrypted backup

Access control

Why the Keychain?

Protected with the user passcode

Protected with the device secret

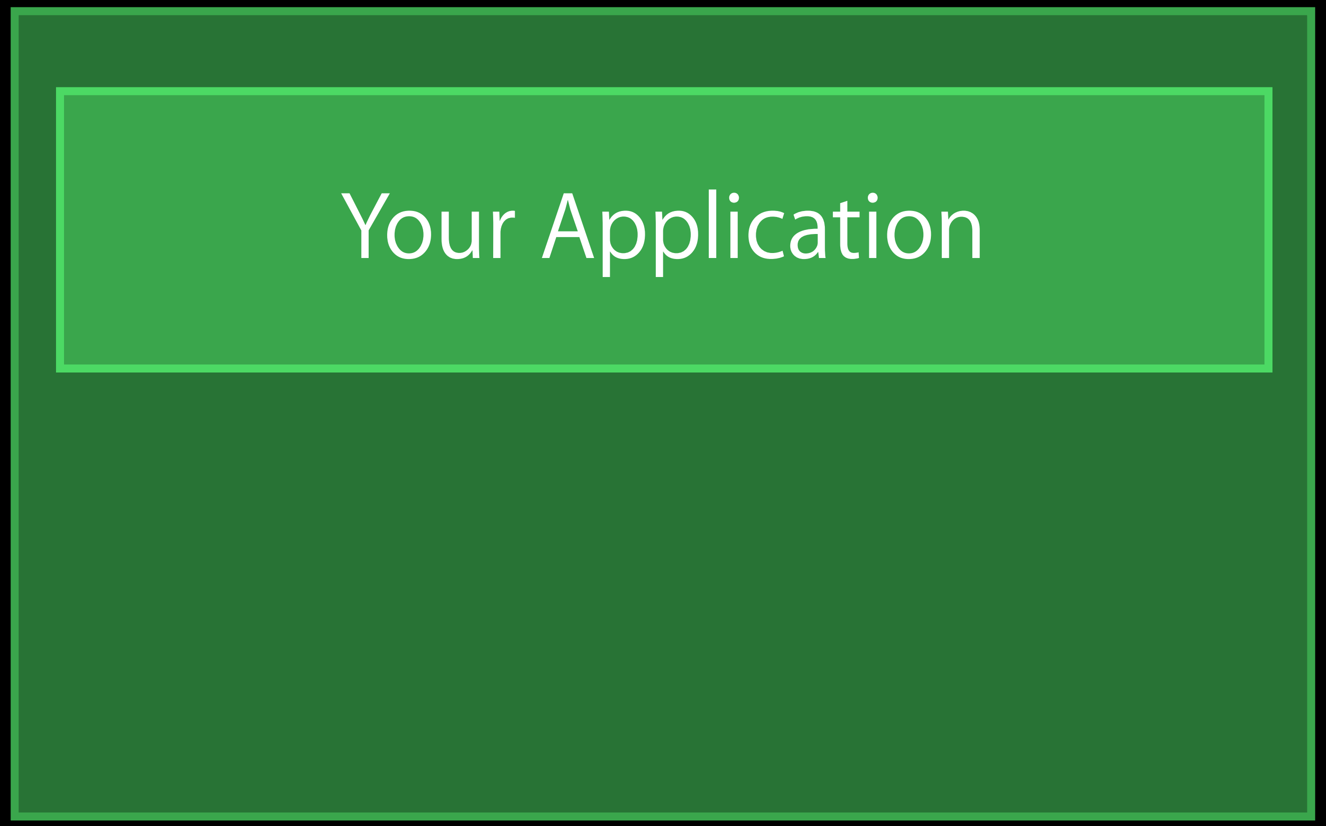
Protect secrets at rest

Encrypted backup

Access control

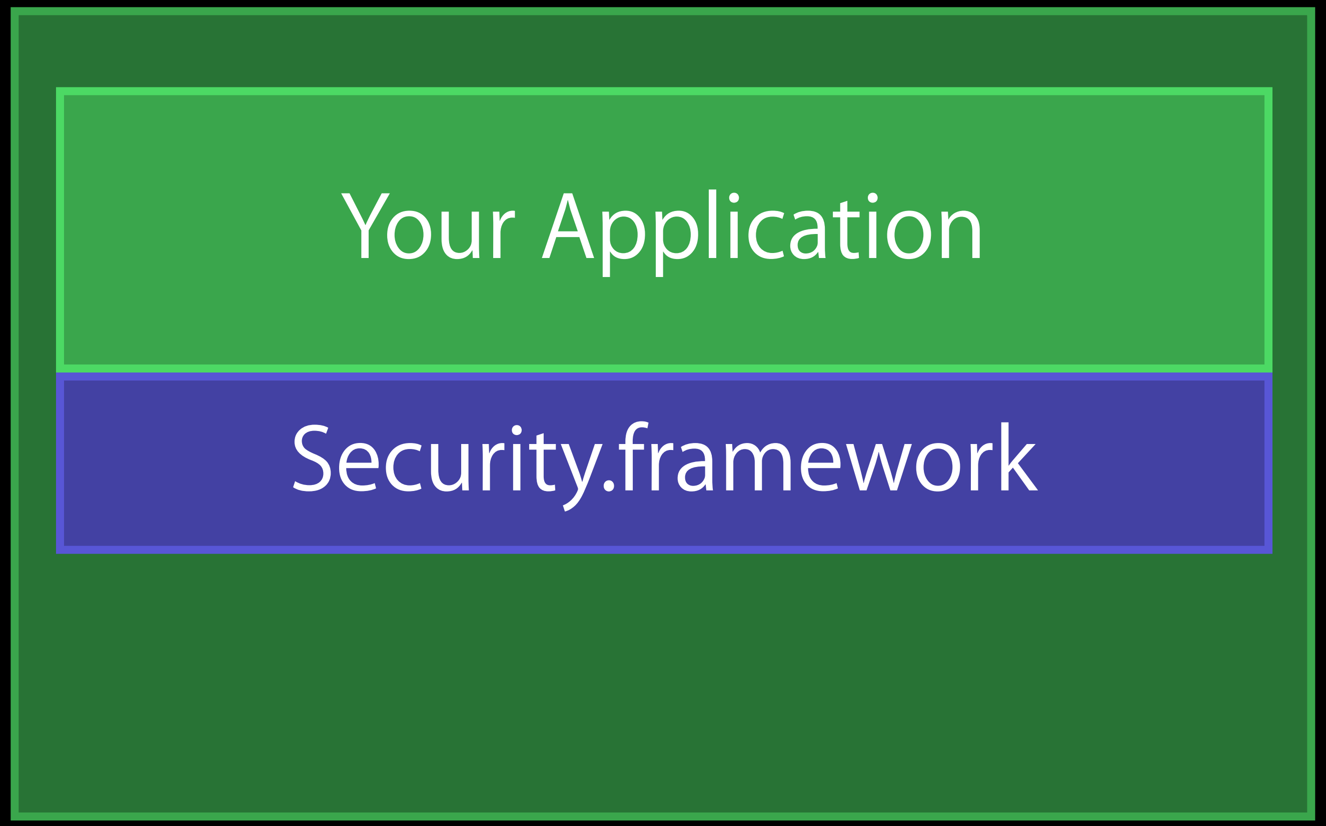
Keychain sync

Keychain Interactions

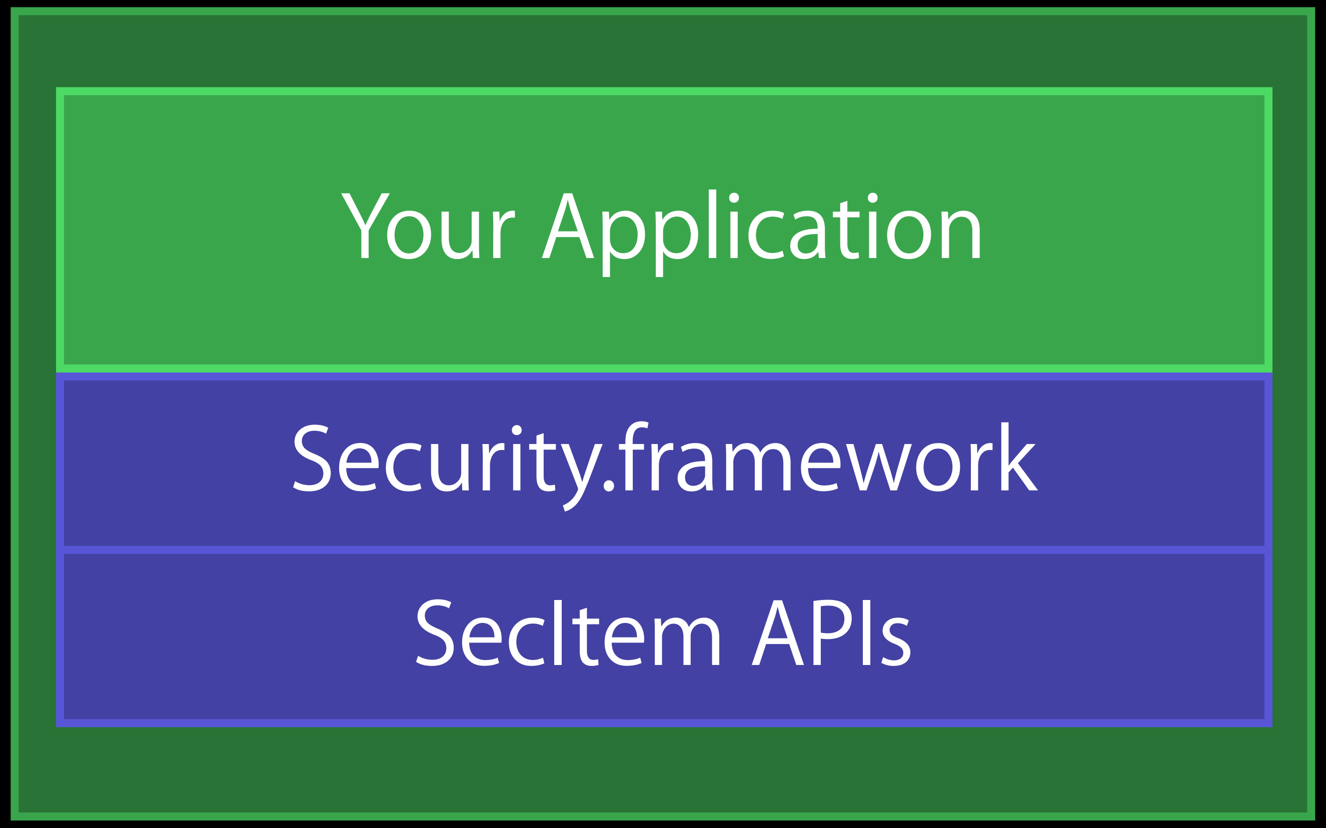


Your Application

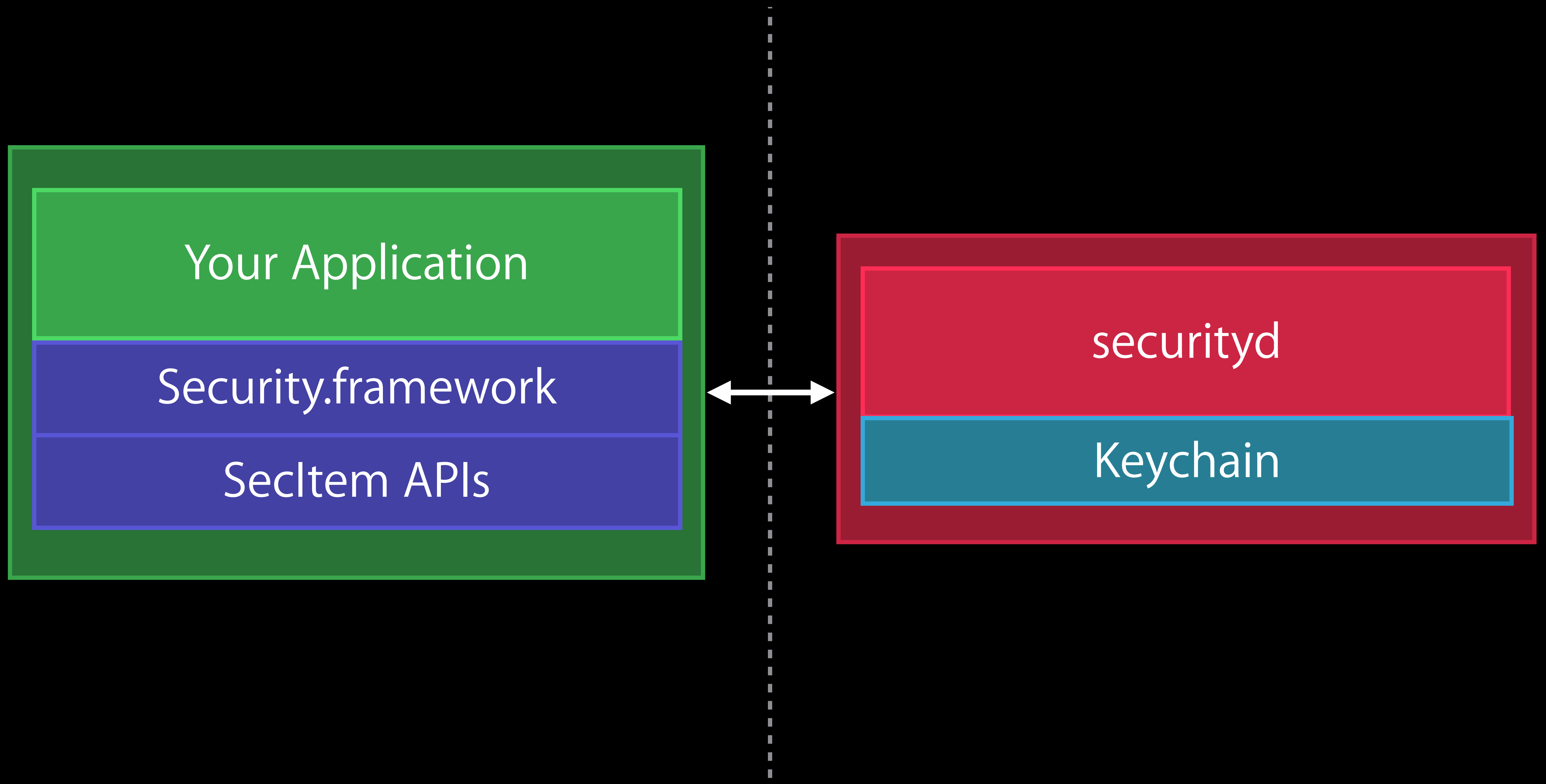
Keychain Interactions



Keychain Interactions



Keychain Interactions



Your Application

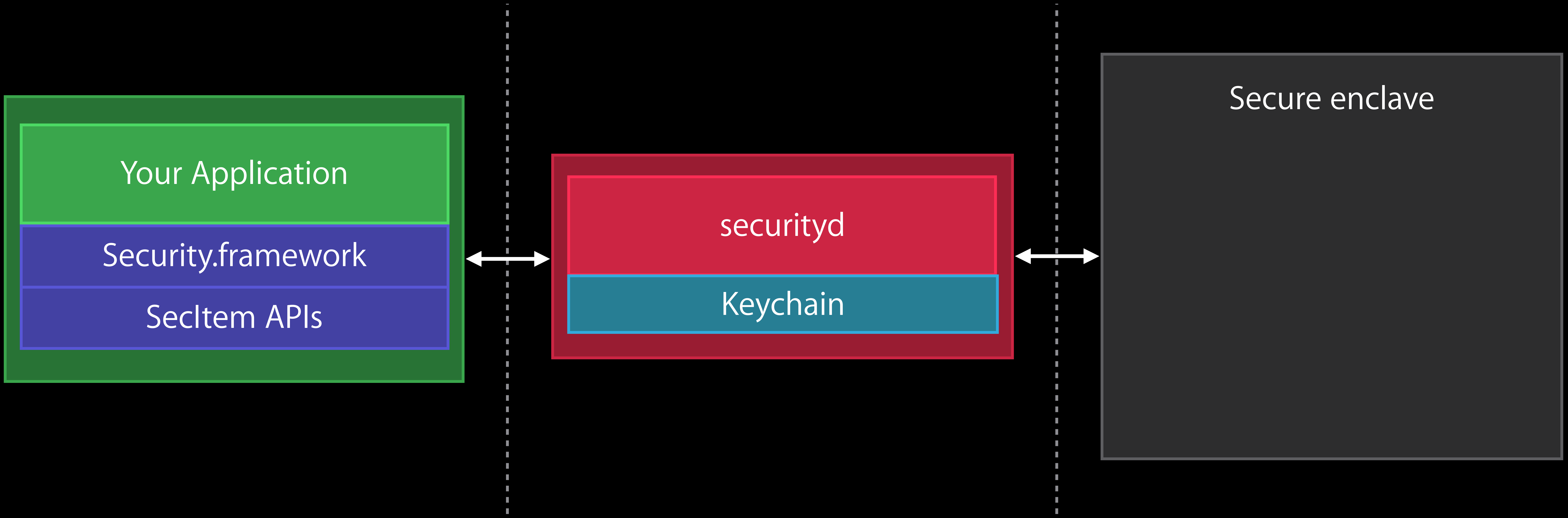
Security.framework

SecItem APIs

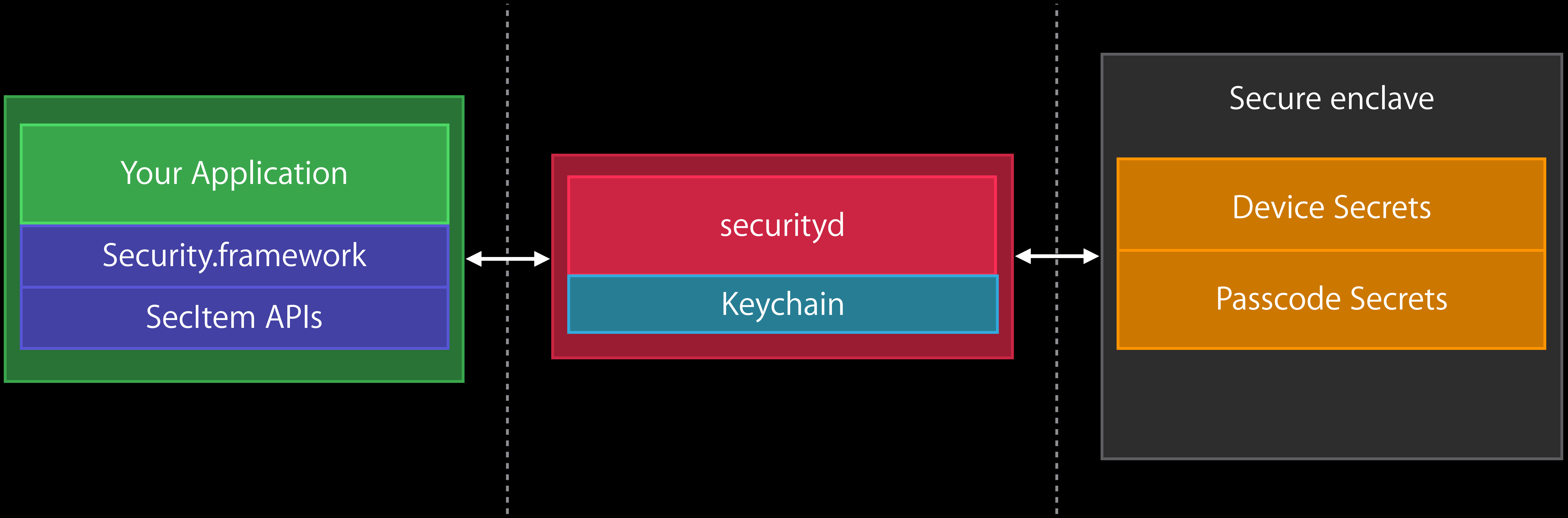
securityd

Keychain

Keychain Interactions



Keychain Interactions



Your Application

Security.framework

SecItem APIs

securityd

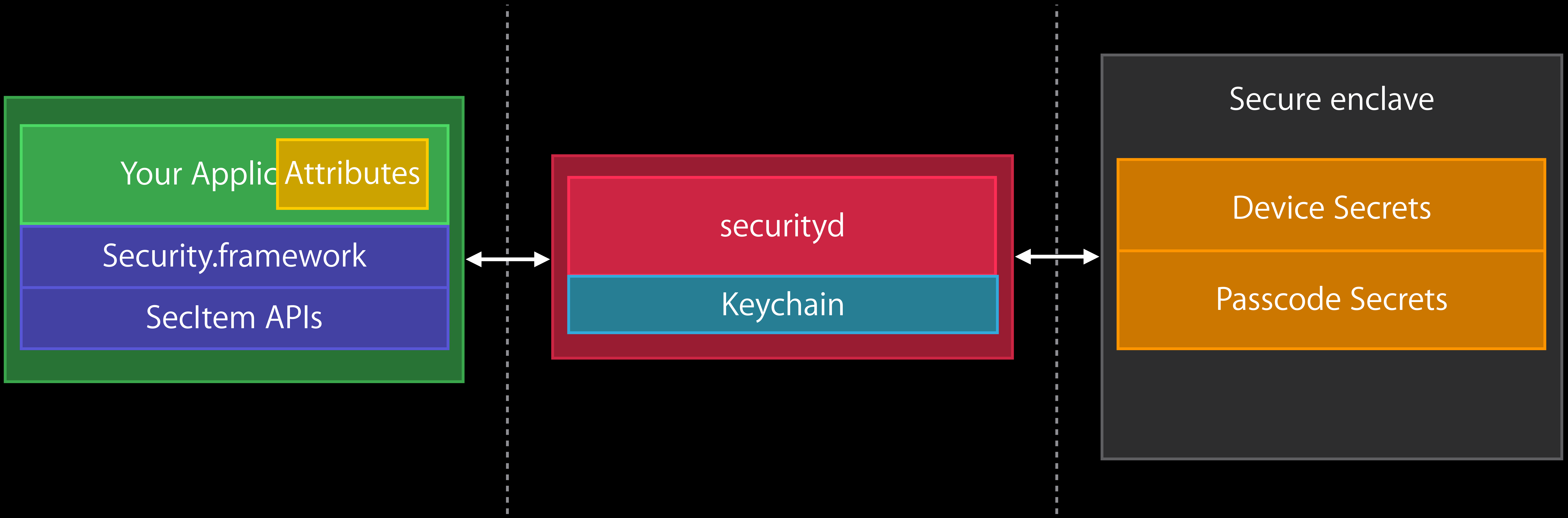
Keychain

Secure enclave

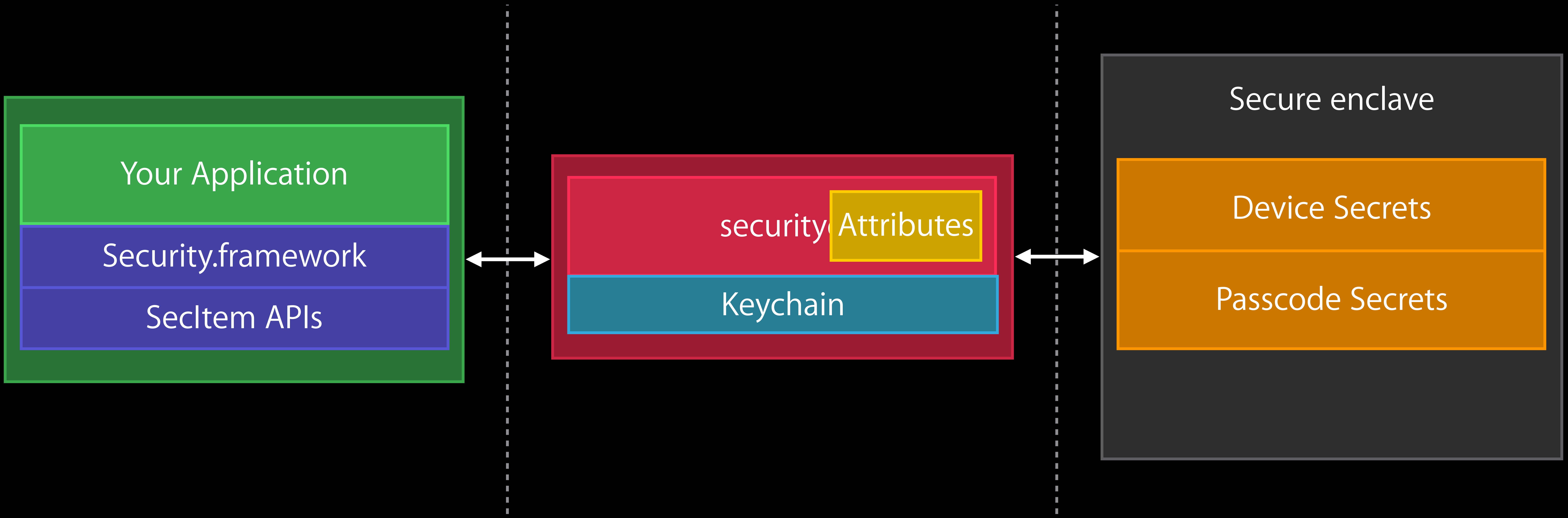
Device Secrets

Passcode Secrets

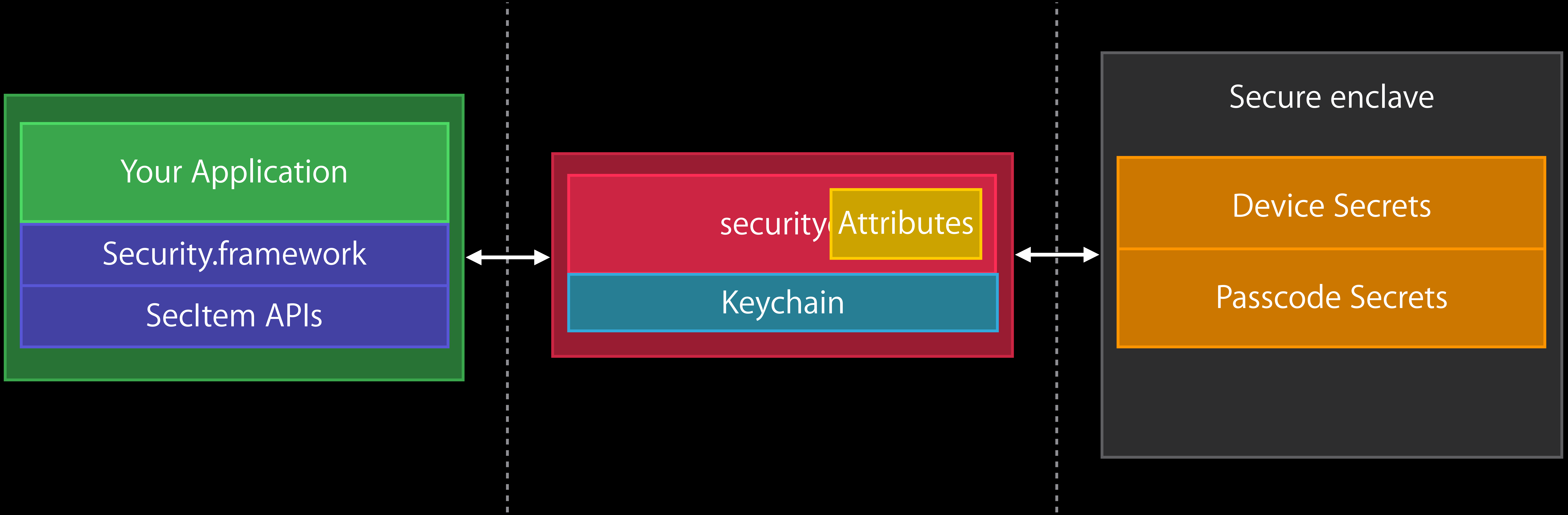
Keychain Interactions



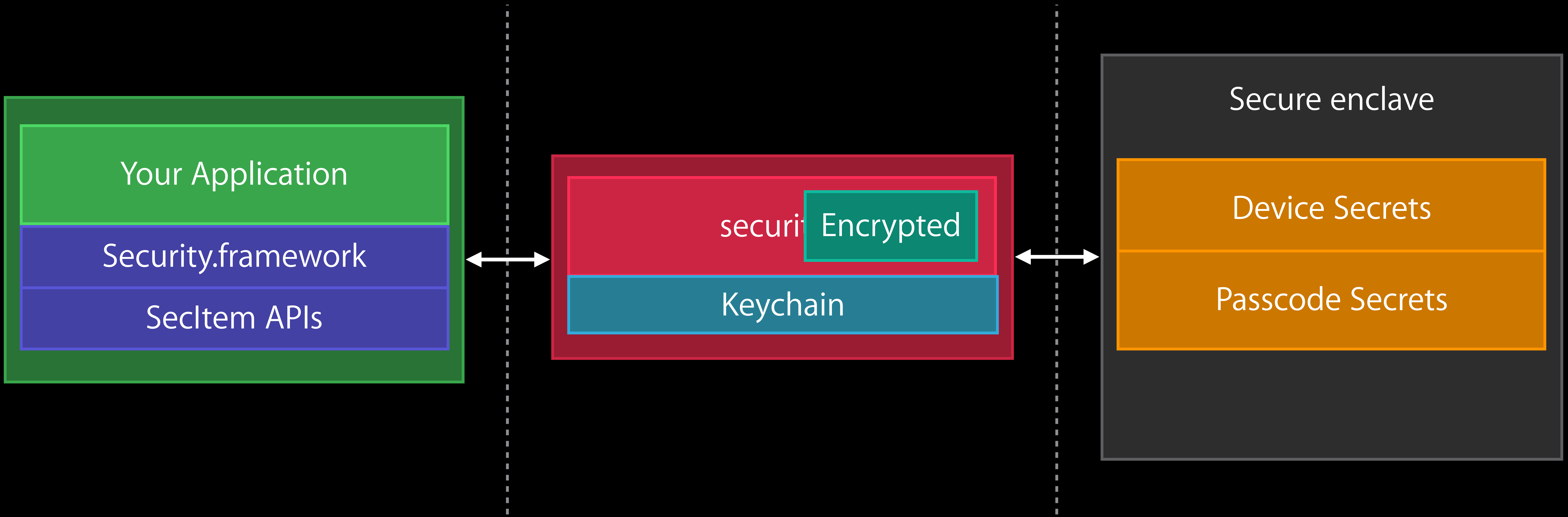
Keychain Interactions



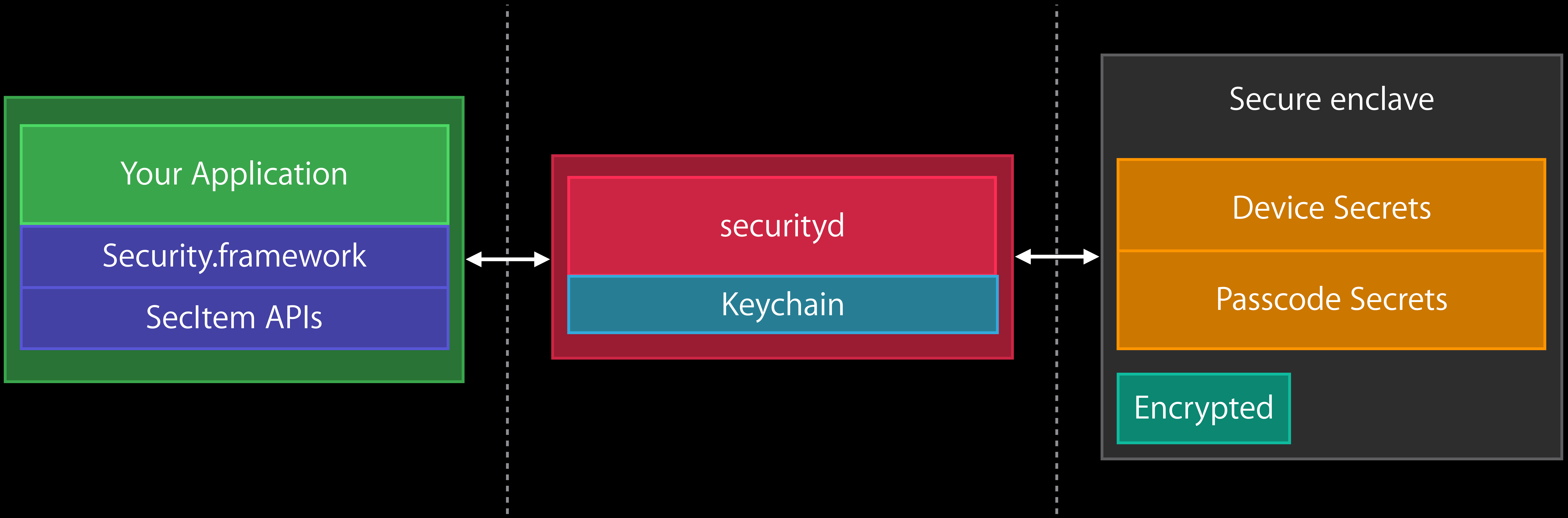
Keychain Interactions



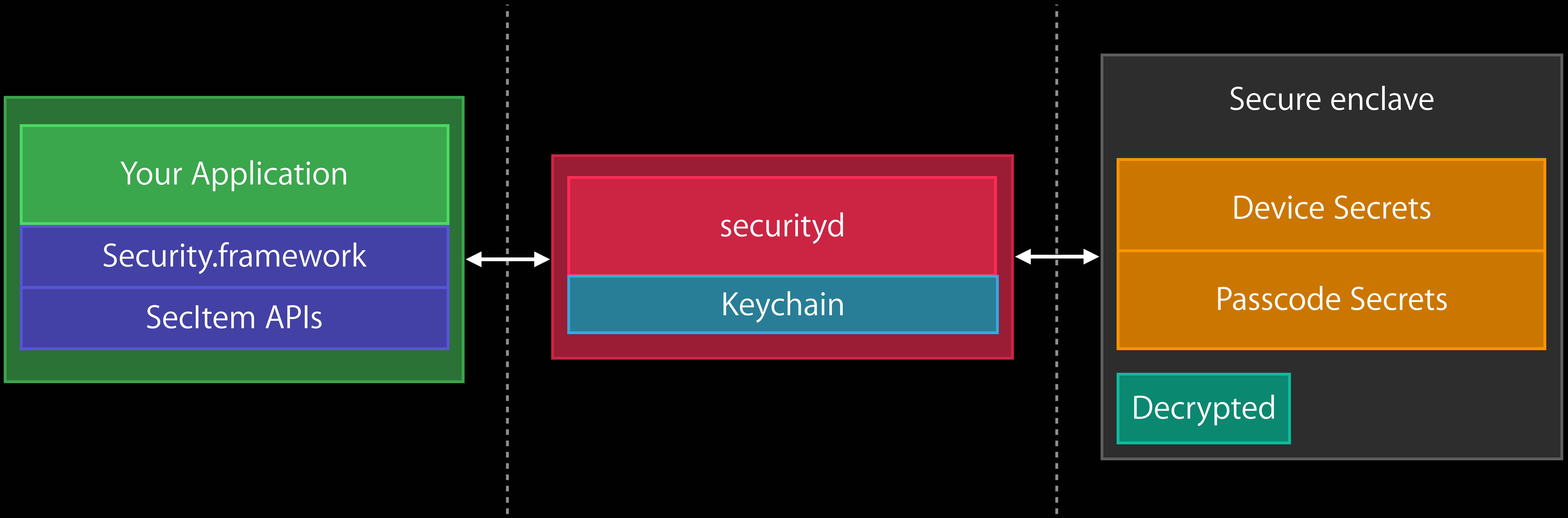
Keychain Interactions



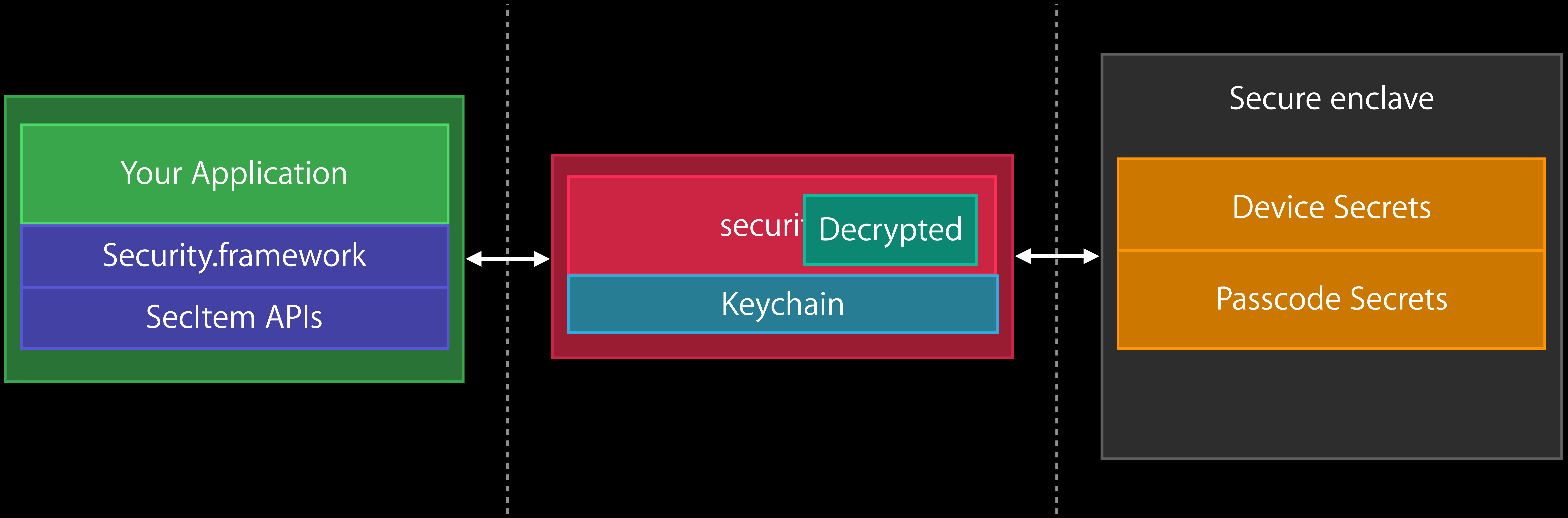
Keychain Interactions



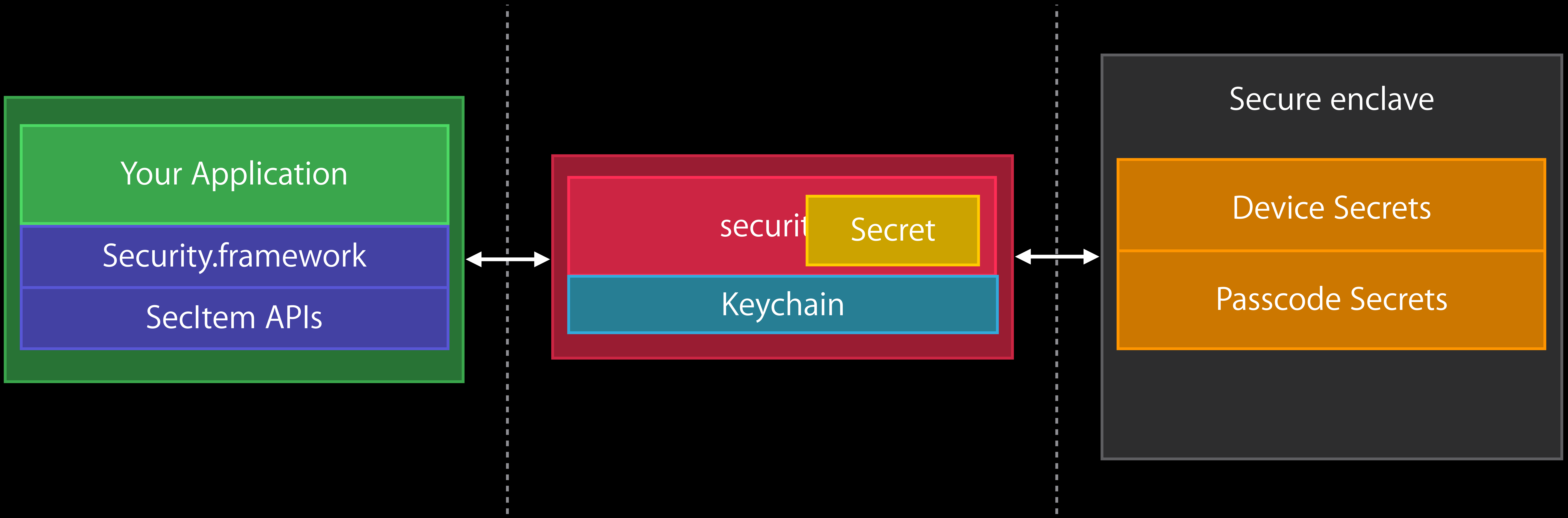
Keychain Interactions



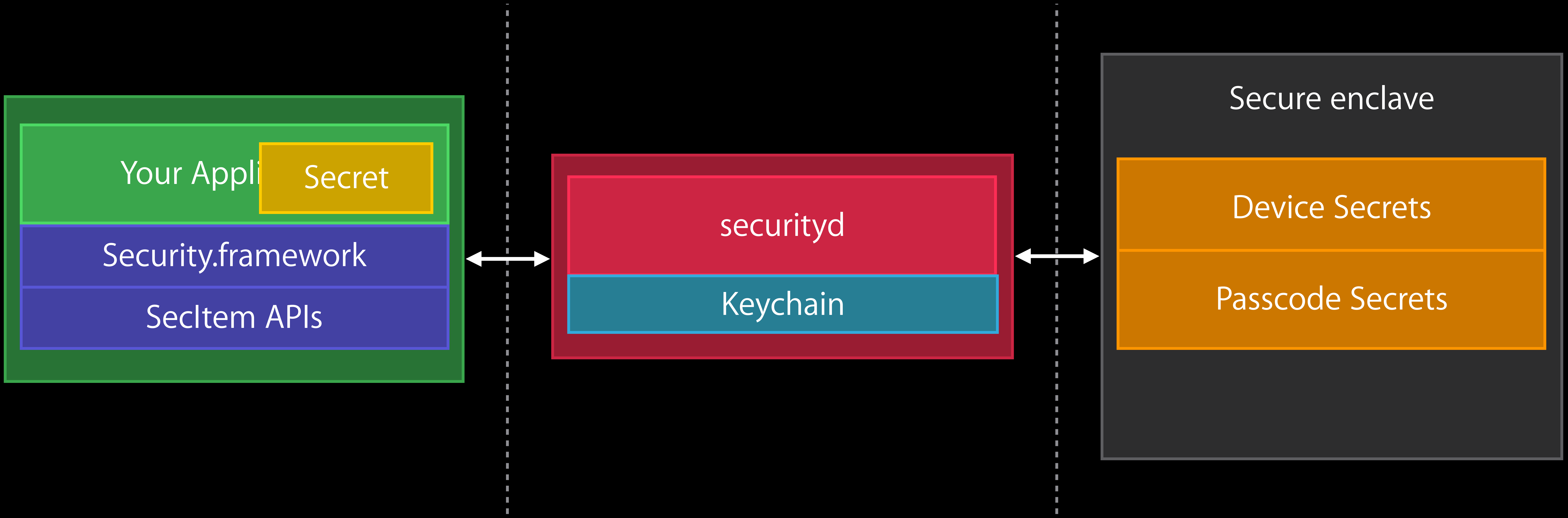
Keychain Interactions



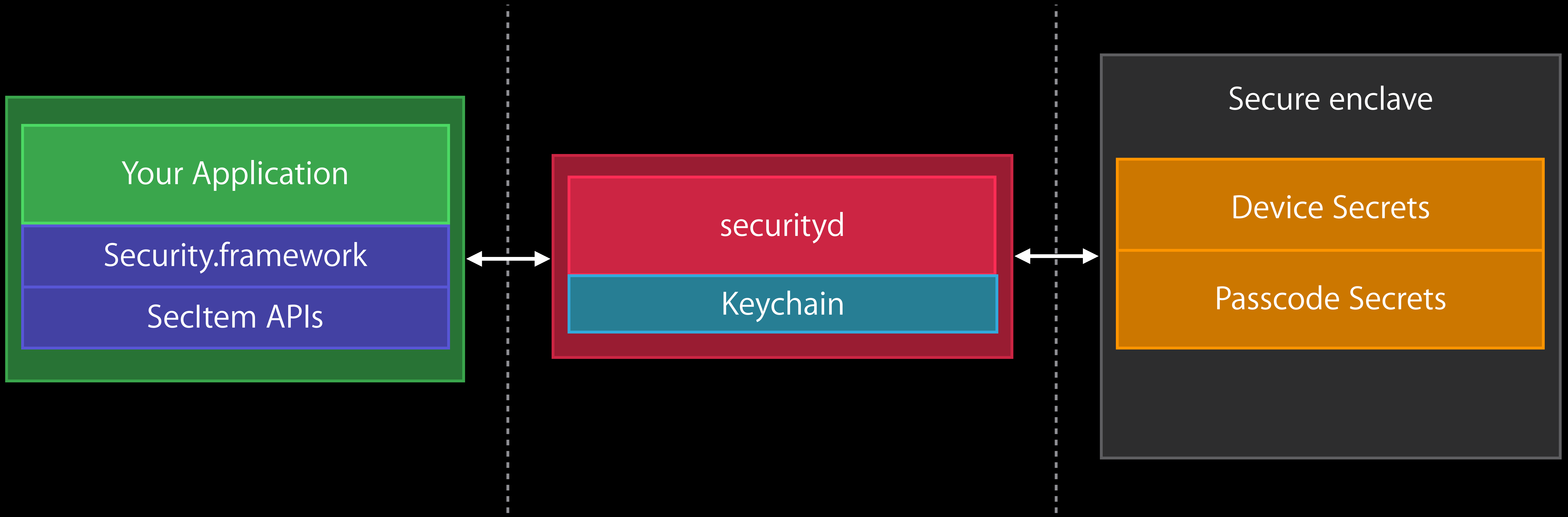
Keychain Interactions



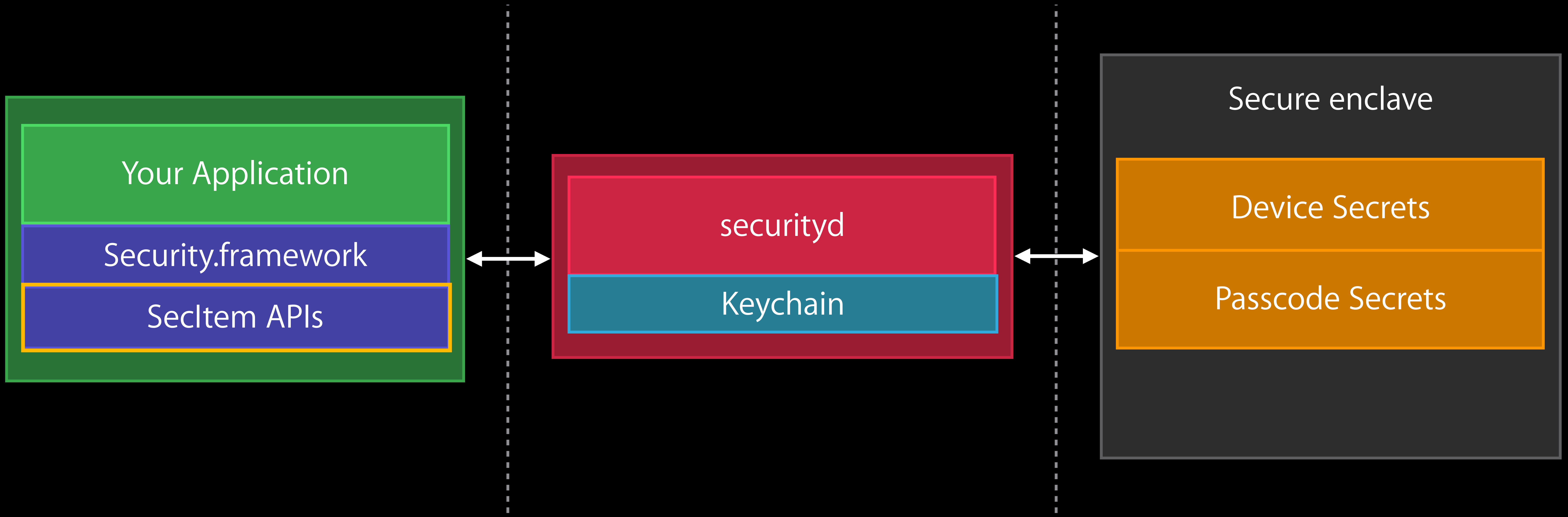
Keychain Interactions



Keychain Interactions



Keychain Interactions



The Keychain in a Nutshell

Item creation

```
NSData* secret = [@"top secret" dataWithEncoding:NSUTF8StringEncoding];
NSDictionary* attributes = @{
     (id)kSecClass: (id)kSecClassGenericPassword,
     (id)kSecAttrService: @"myservice",
     (id)kSecAttrAccount: @"account name here",
     (id)kSecValueData: secret,
};
OSStatus status = SecItemAdd((CFDictionaryRef) attributes, NULL);
```

Cannot create duplicate items

The Keychain in a Nutshell

Item creation

```
NSData* secret = [@"top secret" dataWithEncoding:NSUTF8StringEncoding];
NSDictionary* attributes = @{
     (id)kSecClass: (id)kSecClassGenericPassword,
     (id)kSecAttrService: @"myservice",
     (id)kSecAttrAccount: @"account name here",
     (id)kSecValueData: secret,
};
OSStatus status = SecItemAdd((CFDictionaryRef) attributes, NULL);
```

Cannot create duplicate items

The Keychain in a Nutshell

Item creation

```
NSData* secret = [@"top secret" dataWithEncoding:NSUTF8StringEncoding];
NSDictionary* attributes = @{
     (id)kSecClass: (id)kSecClassGenericPassword,
     (id)kSecAttrService: @"myservice",
     (id)kSecAttrAccount: @"account name here",
     (id)kSecValueData: secret,
};
OSStatus status = SecItemAdd((CFDictionaryRef) attributes, NULL);
```

Cannot create duplicate items

The Keychain in a Nutshell

Item creation

```
NSData* secret = [@"top secret" dataWithEncoding:NSUTF8StringEncoding];
NSDictionary* attributes = @{
     (id)kSecClass: (id)kSecClassGenericPassword,
     (id)kSecAttrService: @"myservice",
     (id)kSecAttrAccount: @"account name here",
     (id)kSecValueData: secret,
};
OSStatus status = SecItemAdd((CFDictionaryRef) attributes, NULL);
```

Cannot create duplicate items

The Keychain in a Nutshell

Item lookup

```
NSDictionary* query = @{
     (id)kSecClass: (id)kSecClassGenericPassword,
     (id)kSecAttrService: @"myservice",
     (id)kSecAttrAccount: @"account name here",
     (id)kSecReturnData: @YES,
};
NSData *data = NULL;
OSStatus status = SecItemCopyMatching((CFDictionaryRef)query,
                                       (CFTypeRef*)&data);
```

The Keychain in a Nutshell

Item lookup

```
NSDictionary* query = @{
     (id)kSecClass: (id)kSecClassGenericPassword,
     (id)kSecAttrService: @"myservice",
     (id)kSecAttrAccount: @"account name here",
     (id)kSecReturnData: @YES,
};
NSData *data = NULL;
OSStatus status = SecItemCopyMatching((CFDictionaryRef)query,
                                      (CFTypeRef*)&data);
```

The Keychain in a Nutshell

Item lookup

```
NSDictionary* query = @{
     (id)kSecClass: (id)kSecClassGenericPassword,
     (id)kSecAttrService: @"myservice",
     (id)kSecAttrAccount: @"account name here",
     (id)kSecReturnData: @YES,
};
NSData *data = NULL;
OSStatus status = SecItemCopyMatching((CFDictionaryRef)query,
                                      (CFTypeRef*)&data);
```

The Keychain in a Nutshell

Item Update/Delete

```
NSDictionary* query = @{
     (id)kSecClass: (id)kSecClassGenericPassword,
     (id)kSecAttrService: @"myservice",
     (id)kSecAttrAccount: @"account name here",
};
NSDictionary* changes = @{ ... };
      OSStatus status = SecItemUpdate((CFDictionaryRef)query,
     (CFDictionaryRef)changes);
OSStatus status = SecItemDelete((CFDictionaryRef)query);
```

The Keychain in a Nutshell

Item Update/Delete

```
NSDictionary* query = @{
     (id)kSecClass: (id)kSecClassGenericPassword,
     (id)kSecAttrService: @"myservice",
     (id)kSecAttrAccount: @"account name here",
};
NSDictionary* changes = @{ ... };
      OSStatus status = SecItemUpdate((CFDictionaryRef)query,
     (CFDictionaryRef)changes);
OSStatus status = SecItemDelete((CFDictionaryRef)query);
```

The Keychain in a Nutshell

Item Update/Delete

```
NSDictionary* query = @{
     (id)kSecClass: (id)kSecClassGenericPassword,
     (id)kSecAttrService: @"myservice",
     (id)kSecAttrAccount: @"account name here",
};
NSDictionary* changes = @{ ... };
      OSStatus status = SecItemUpdate((CFDictionaryRef)query,
     (CFDictionaryRef)changes);
OSStatus status = SecItemDelete((CFDictionaryRef)query);
```

The Keychain in a Nutshell

Item Update/Delete

```
NSDictionary* query = @{
     (id)kSecClass: (id)kSecClassGenericPassword,
     (id)kSecAttrService: @"myservice",
     (id)kSecAttrAccount: @"account name here",
};
NSDictionary* changes = @{ ... };
      OSStatus status = SecItemUpdate((CFDictionaryRef)query,
     (CFDictionaryRef)changes);
OSStatus status = SecItemDelete((CFDictionaryRef)query);
```

Use Update to modify items, not Delete and Add

Keychain Workflow

(Pseudo code)

Keychain Workflow

(Pseudo code)

```
NSData* password = nil;
if (SecItemCopyMatching(..., &password) == noErr) {
    if (password works) {
        // great!
```

Keychain Workflow

(Pseudo code)

```
NSData* password = nil;
if (SecItemCopyMatching(..., &password) == noErr) {
    if (password works) {
        // great!




} else {
    password = get from user; if (password works) {
    SecItemAdd(...);   // save it for next time
    }
}
```

Keychain Workflow

(Pseudo code)

```
NSData* password = nil;
if (SecItemCopyMatching(..., &password) == noErr) {
    if (password works) {
        // great!
    } else {
        password = get a better one;
        if (that password worked better) {
            SecItemUpdate(...);
        }
    }
} else {
    password = get from user; if (password works) {
    SecItemAdd(...);   // save it for next time
    }
}
```

Handling Retrieved Secrets

Use and purge

Do not keep secrets in memory

Do not save or send

Access Control Roadmap

Access Control Roadmap

	OS X
Application	kSecAttrAccess

Access Control Roadmap

	OS X	iOS
Application	kSecAttrAccess	
Entitlements		kSecAttrAccessGroup

Access Control Roadmap

	OS X	iOS
Application	kSecAttrAccess	
Entitlements		kSecAttrAccessGroup
When		kSecAttrAccessible

Access Control Roadmap

	OS X	iOS
Application	kSecAttrAccess	
Entitlements		kSecAttrAccessGroup
When		kSecAttrAccessible
Authentication		kSecAttrAccessControl

Access Control Roadmap

	OS X	iOS	OS X kSecAttrSynchronizable
Application	kSecAttrAccess		
Entitlements		kSecAttrAccessGroup	kSecAttrAccessGroup
When		kSecAttrAccessible	kSecAttrAccessible
Authentication		kSecAttrAccessControl	kSecAttrAccessControl

Access Control Roadmap

	OS X	iOS	OS X kSecAttrSynchronizable
Application	kSecAttrAccess		
Entitlements		kSecAttrAccessGroup	kSecAttrAccessGroup
When		kSecAttrAccessible	kSecAttrAccessible
Authentication		kSecAttrAccessControl	kSecAttrAccessControl

Keychain Item Access

```
(id)kSecAttrAccessible:
```

Keychain Item Access

```
(id)kSecAttrAccessible:
kSecAttrAccessibleWhenUnlocked
```

Keychain Item Access

```
(id)kSecAttrAccessible:
kSecAttrAccessibleWhenUnlocked
kSecAttrAccessibleAfterFirstUnlock
```

Keychain Item Access

```
(id)kSecAttrAccessible:
kSecAttrAccessibleWhenUnlocked
kSecAttrAccessibleAfterFirstUnlock
kSecAttrAccessibleWhenUnlockedThisDeviceOnly
kSecAttrAccessibleAfterFirstUnlockThisDeviceOnly
```

Passcode Set?

```
(id)kSecAttrAccessible:
kSecAttrAccessibleWhenPasscodeSetThisDeviceOnly
```

Passcode Set?

```
(id)kSecAttrAccessible:
kSecAttrAccessibleWhenPasscodeSetThisDeviceOnly
```

Available if a passcode set

Passcode Set?

```
(id)kSecAttrAccessible:
kSecAttrAccessibleWhenPasscodeSetThisDeviceOnly
```

Available if a passcode set

Removing the device passcode

Passcode Set?

```
(id)kSecAttrAccessible:
kSecAttrAccessibleWhenPasscodeSetThisDeviceOnly
```

Available if a passcode set

Removing the device passcode

Will not sync

Passcode Set?

```
(id)kSecAttrAccessible:
kSecAttrAccessibleWhenPasscodeSetThisDeviceOnly
```

Available if a passcode set

Removing the device passcode

Will not sync

Not backed up

Keychain ACLs and Touch ID

Libor Sykora
Core OS Security Engineering

Introduction

Keychain ACLs and Touch ID

Introduction

Keychain ACLs and Touch ID

Item Access Control Lists (ACLs)

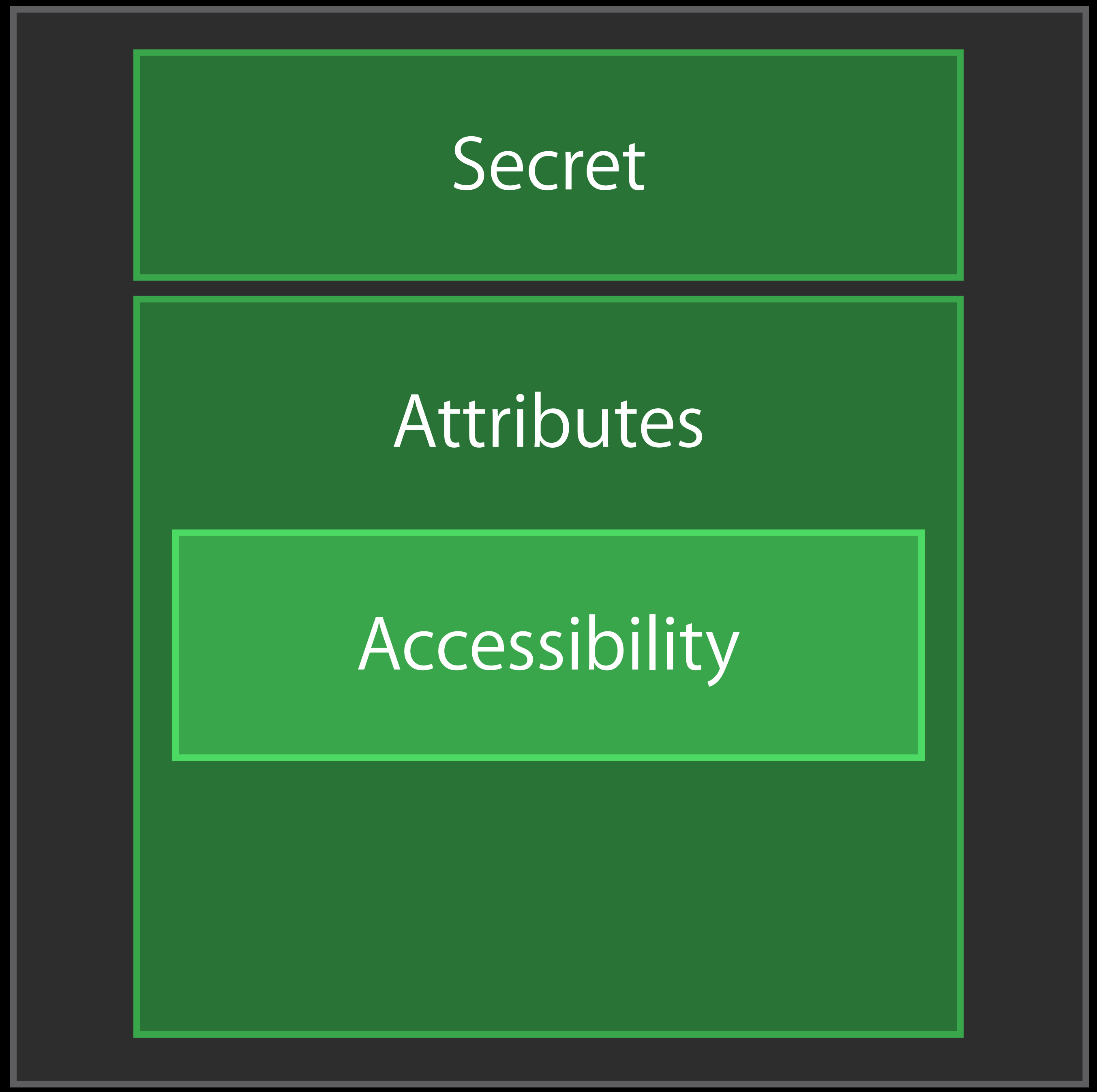
- Accessibility
- Authentication

Introduction

Keychain ACLs and Touch ID

Item Access Control Lists (ACLs)

- Accessibility
- Authentication

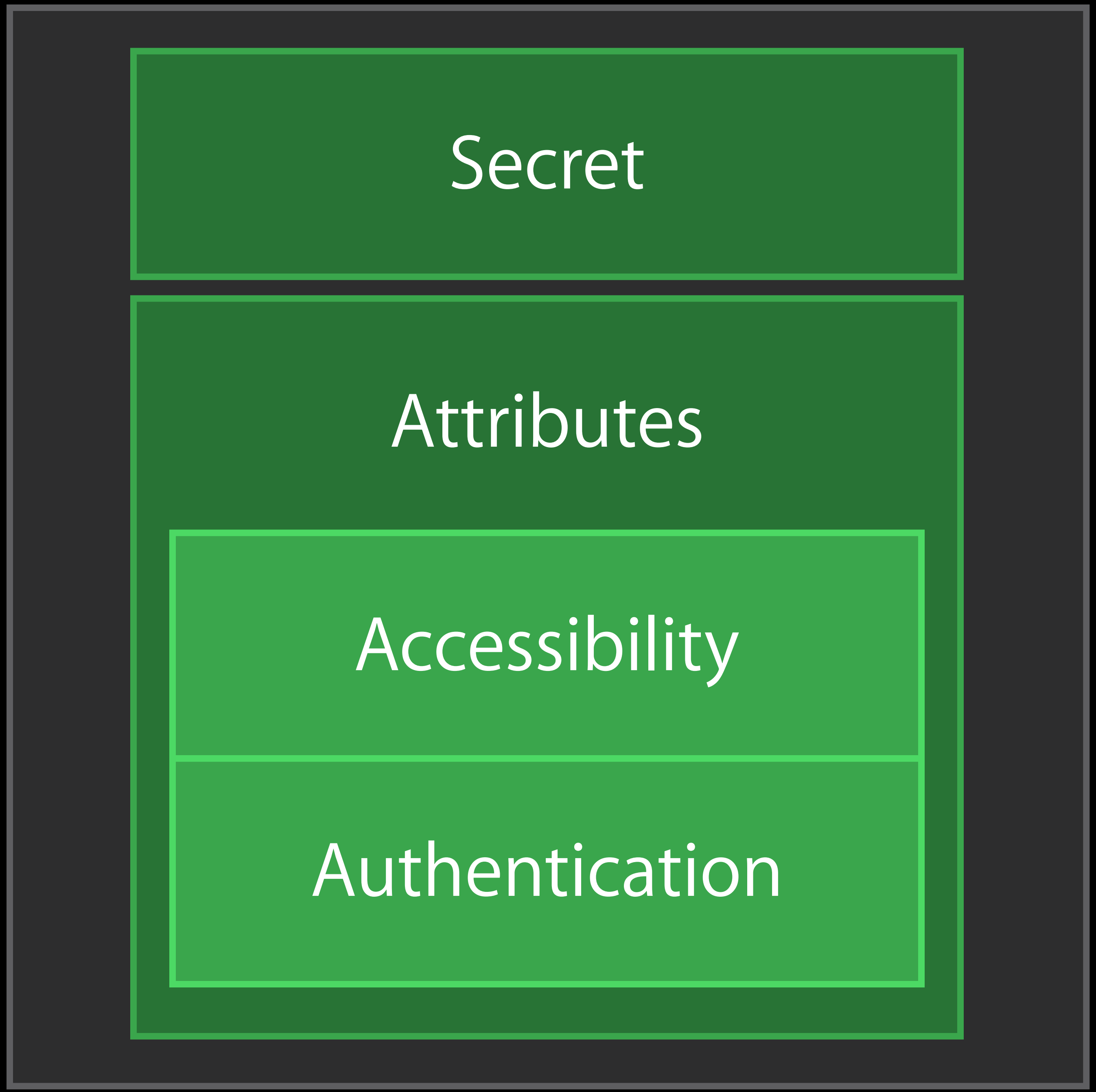


Introduction

Keychain ACLs and Touch ID

Item Access Control Lists (ACLs)

- Accessibility
- Authentication

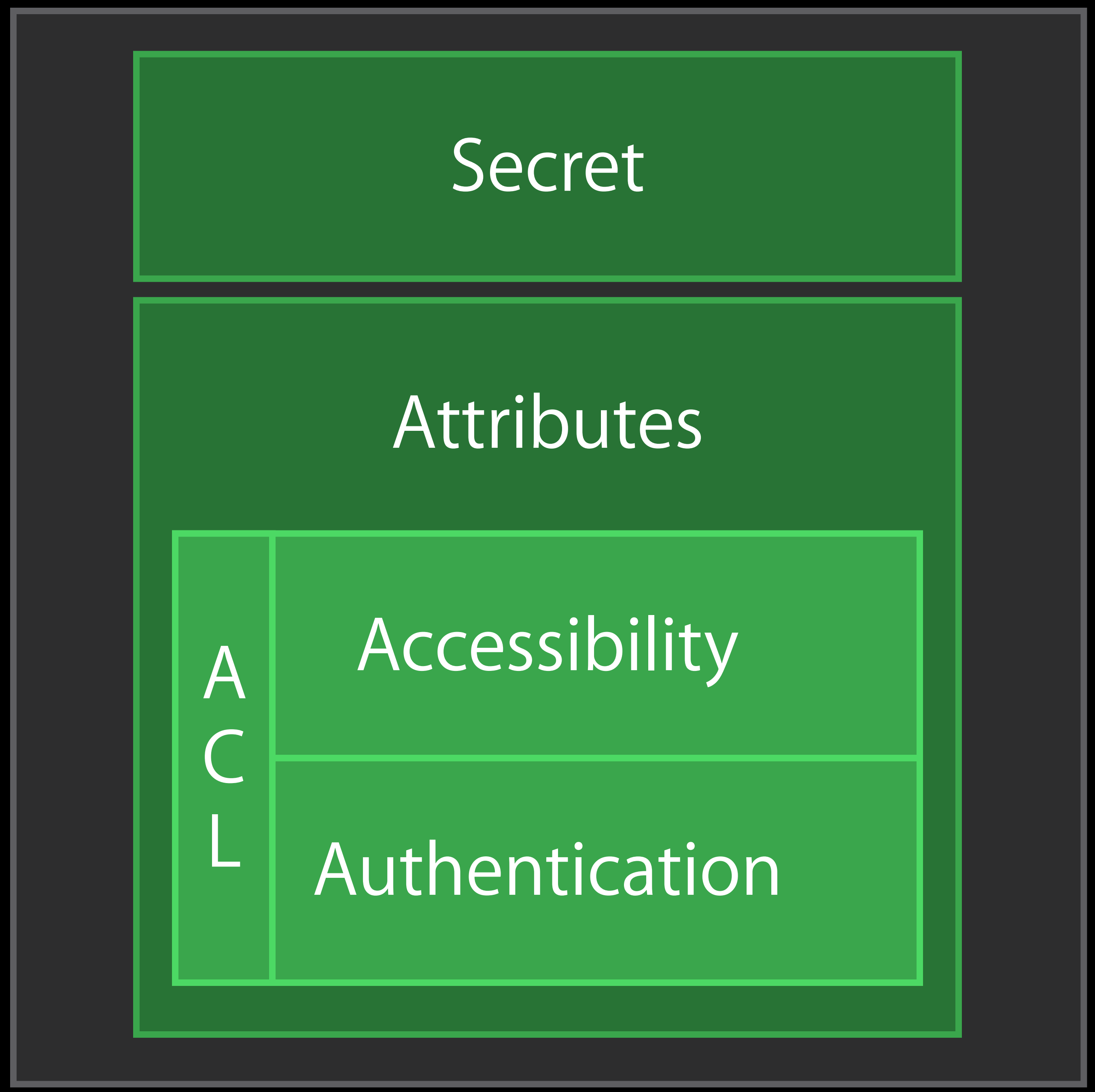


Introduction

Keychain ACLs and Touch ID

Item Access Control Lists (ACLs)

- Accessibility
- Authentication



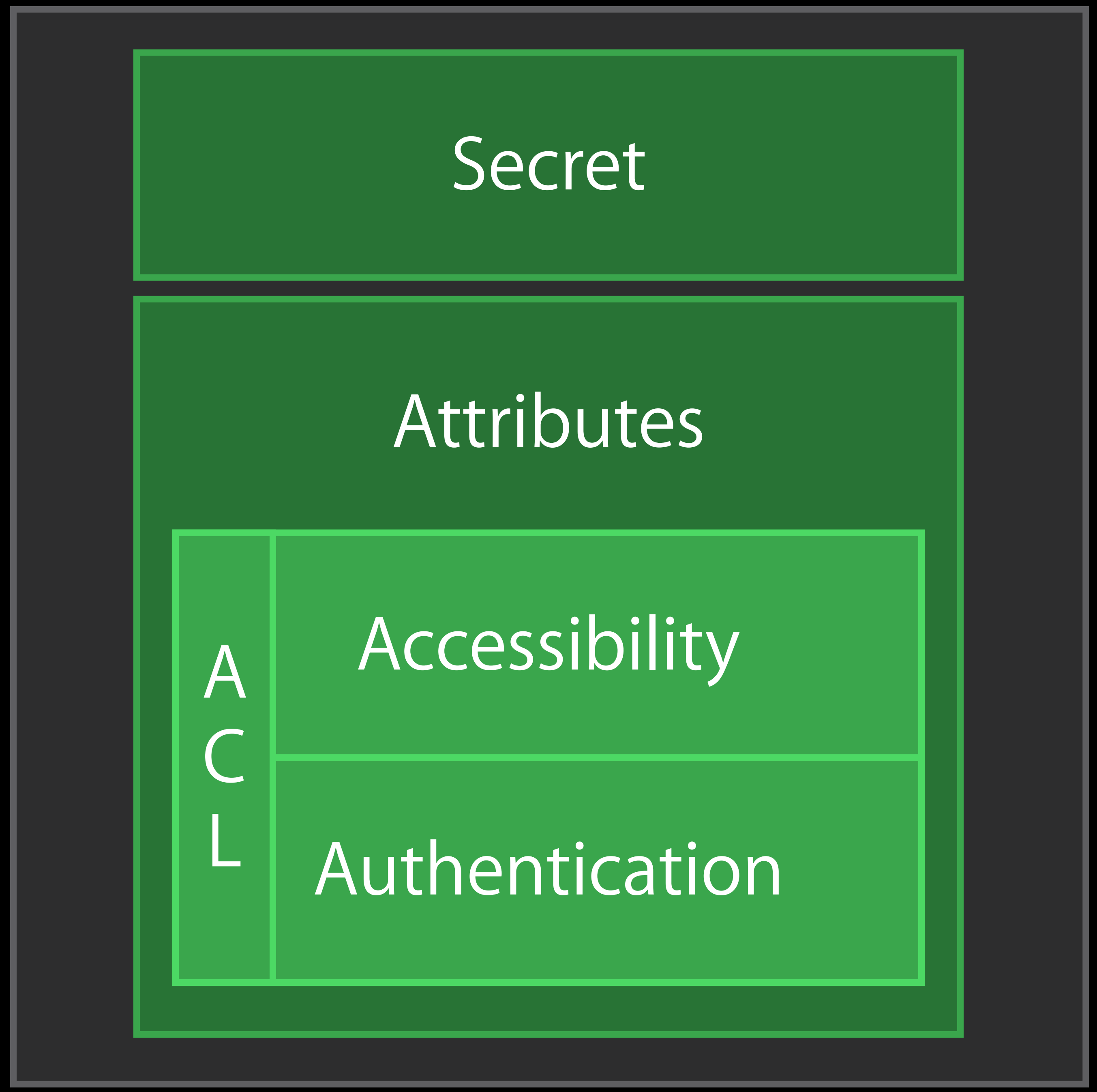
Introduction

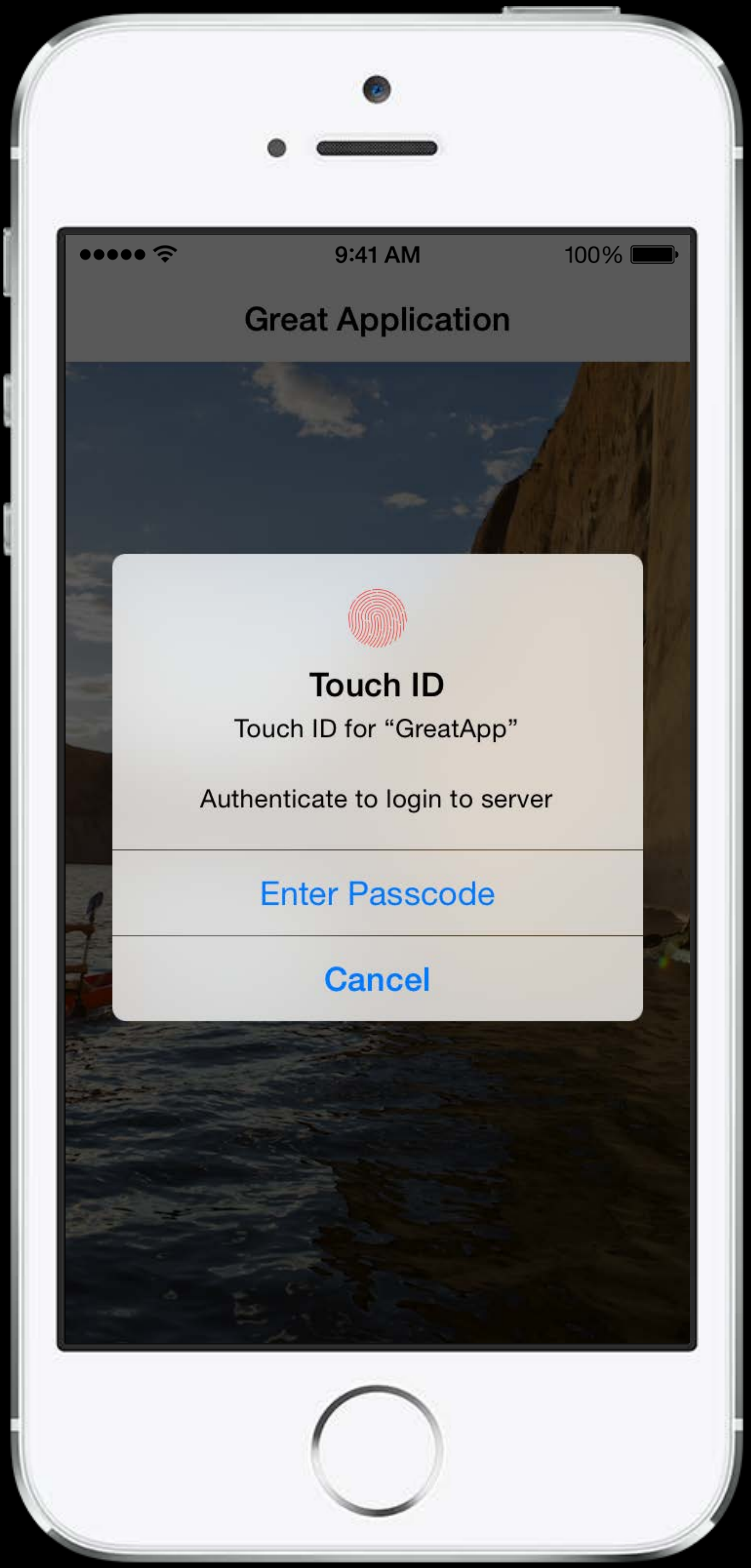
Keychain ACLs and Touch ID

Item Access Control Lists (ACLs)

- Accessibility
- Authentication

Authentication with Touch ID and passcode





Great Application

Touch ID

Touch ID for “GreatApp”

Authenticate to login to server

Keychain ACL

New API for ACLs

Keychain ACL

New API for ACLs

- `(id)kSecAttrAccessControl:`

Keychain ACL

New API for ACLs

- `(id)kSecAttrAccessControl:`
    - `SecAccessControlRef`—ACL

Keychain ACL

New API for ACLs

- `(id)kSecAttrAccessControl:`
  - `SecAccessControlRef`—ACL

ACL defines both authentication and item accessibility

Keychain ACL

New API for ACLs

- `(id)kSecAttrAccessControl:`
  - `SecAccessControlRef`—ACL

ACL defines both authentication and item accessibility

Accessibility

Keychain ACL

New API for ACLs

- `(id)kSecAttrAccessControl:`
  - `SecAccessControlRef`—ACL

ACL defines both authentication and item accessibility

Accessibility

- `kSecAttrAccessible` constants

Keychain ACL

New API for ACLs

- (id)kSecAttrAccessControl:
  - SecAccessControlRef—ACL

ACL defines both authentication and item accessibility

Accessibility

- kSecAttrAccessible constants

Authentication

Keychain ACL

New API for ACLs

- `(id)kSecAttrAccessControl:`
  - `SecAccessControlRef`—ACL

ACL defines both authentication and item accessibility

Accessibility

- `kSecAttrAccessible` constants

Authentication

- Policy

Keychain ACL

New API for ACLs

- `(id)kSecAttrAccessControl:`
  - `SecAccessControlRef`—ACL

ACL defines both authentication and item accessibility

Accessibility

- `kSecAttrAccessible` constants

Authentication

- Policy
- User presence (`kSecAccessControlUserPresence`)

User Presence

Authentication

Policy

Device configuration	Policy evaluation

User Presence

Authentication

Policy

Enforced by OS security domain

Device configuration	Policy evaluation

User Presence

Authentication

Policy

Enforced by OS security domain

Device configuration	Policy evaluation
Device without passcode	

User Presence

Authentication

Policy

Enforced by OS security domain

Device configuration	Policy evaluation
Device without passcode	No access

User Presence

Authentication

Policy

Enforced by OS security domain

Device configuration	Policy evaluation
Device without passcode	No access
Device with passcode	

User Presence

Authentication

Policy

Enforced by OS security domain

Device configuration	Policy evaluation
Device without passcode	No access
Device with passcode	Requires passcode

User Presence

Authentication

Policy

Enforced by OS security domain

Device configuration	Policy evaluation
Device without passcode	No access
Device with passcode	Requires passcode
Device with Touch ID	

User Presence

Authentication

Policy

Enforced by OS security domain

Device configuration	Policy evaluation
Device without passcode	No access
Device with passcode	Requires passcode
Device with Touch ID	Prefers Touch ID

User Presence

Authentication

Policy

Enforced by OS security domain

Device configuration	Policy evaluation	Backup mechanism
Device without passcode	No access	No backup
Device with passcode	Requires passcode	No backup
Device with Touch ID	Prefers Touch ID	Allows passcode

Touch ID

Authentication

Touch ID—Easy to use authentication mechanism

Touch ID

Authentication

Touch ID—Easy to use authentication mechanism

Touch ID operations—Inside secure enclave

Touch ID

Authentication

Touch ID—Easy to use authentication mechanism

Touch ID operations—Inside secure enclave

Keychain keys operations—Inside secure enclave

Touch ID

Authentication

Touch ID—Easy to use authentication mechanism

Touch ID operations—Inside secure enclave

Keychain keys operations—Inside secure enclave

Touch ID policy enforcement

Touch ID

Authentication

Touch ID—Easy to use authentication mechanism

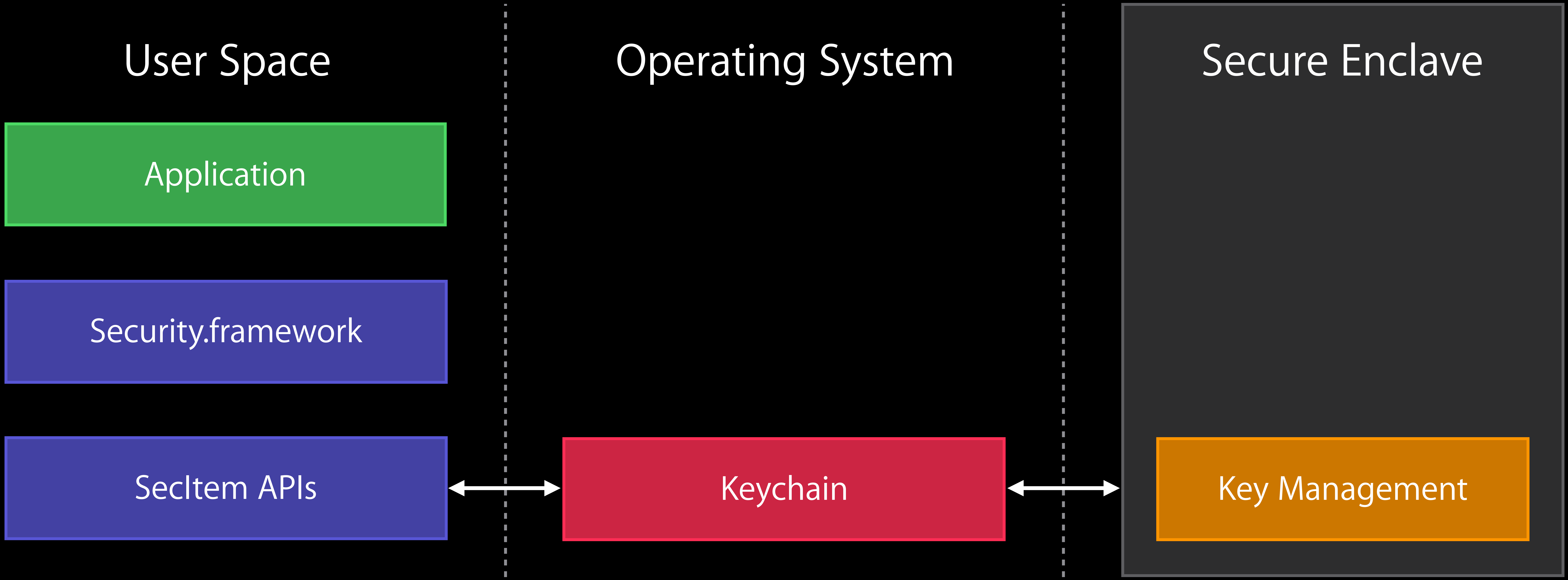
Touch ID operations—Inside secure enclave

Keychain keys operations—Inside secure enclave

Touch ID policy enforcement

Touch ID through LocalAuthentication

Architecture



User Space

Application

Security.framework

SecItem APIs

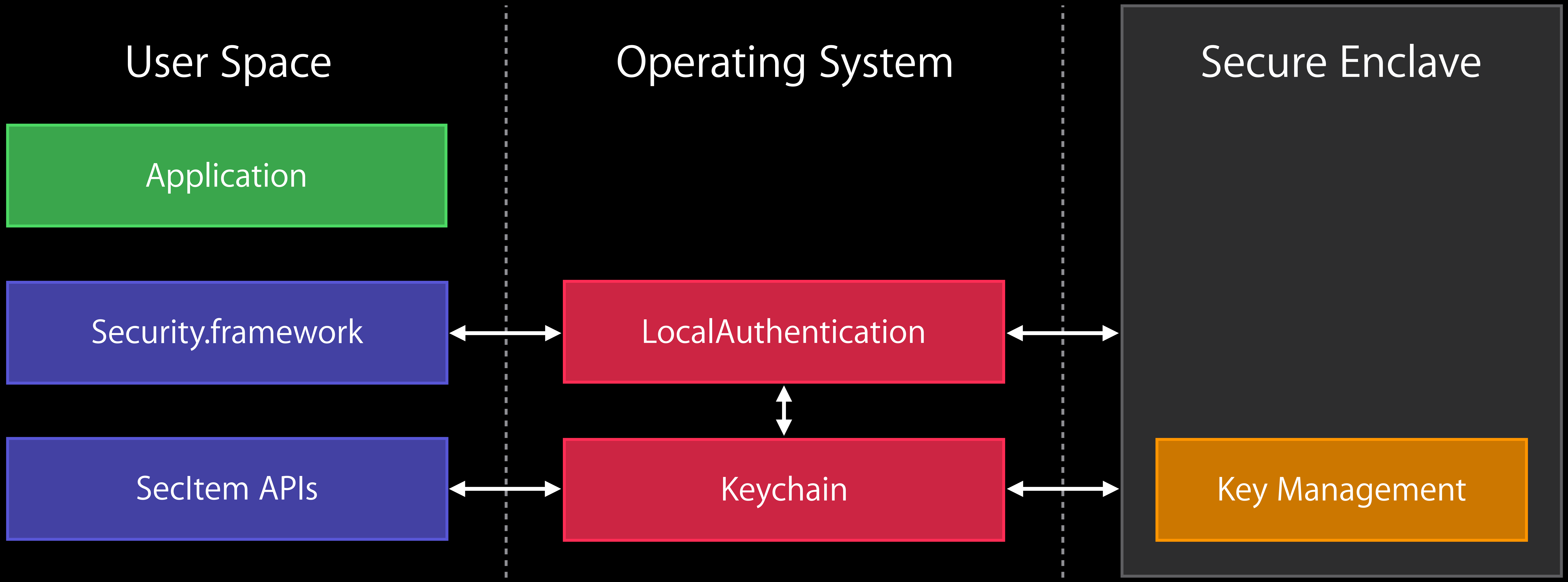
Operating System

Keychain

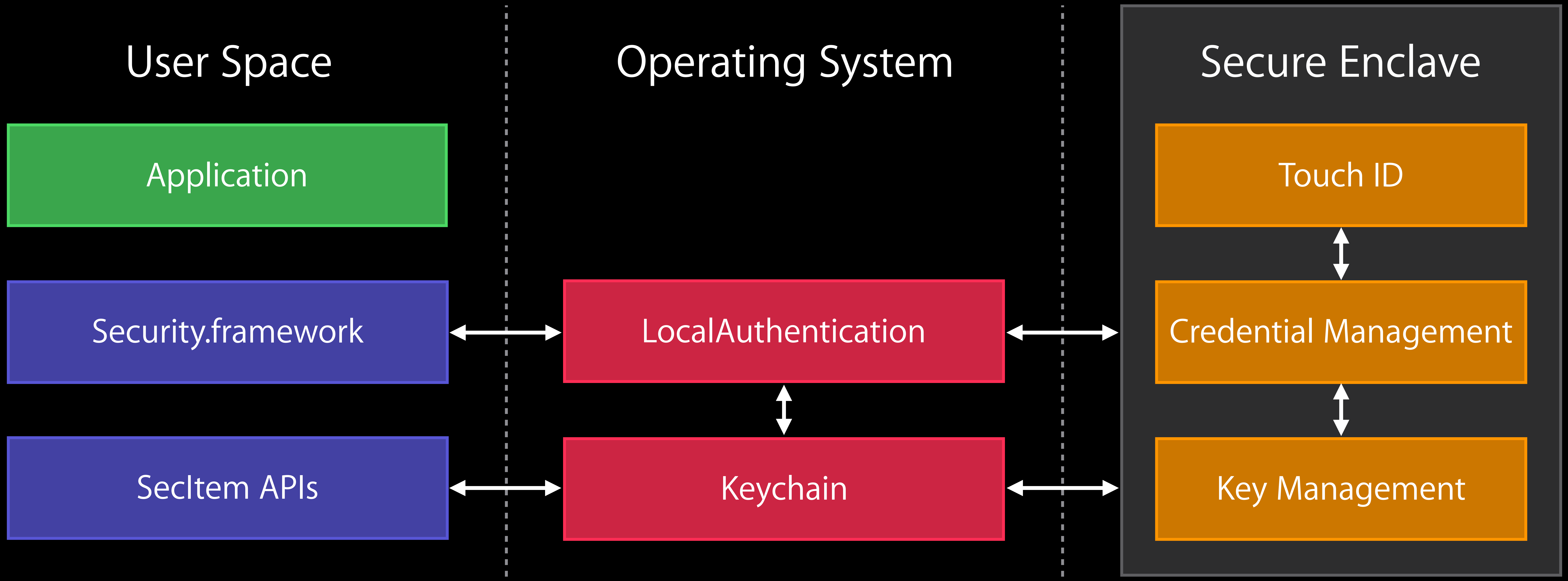
Secure Enclave

Key Management

Architecture

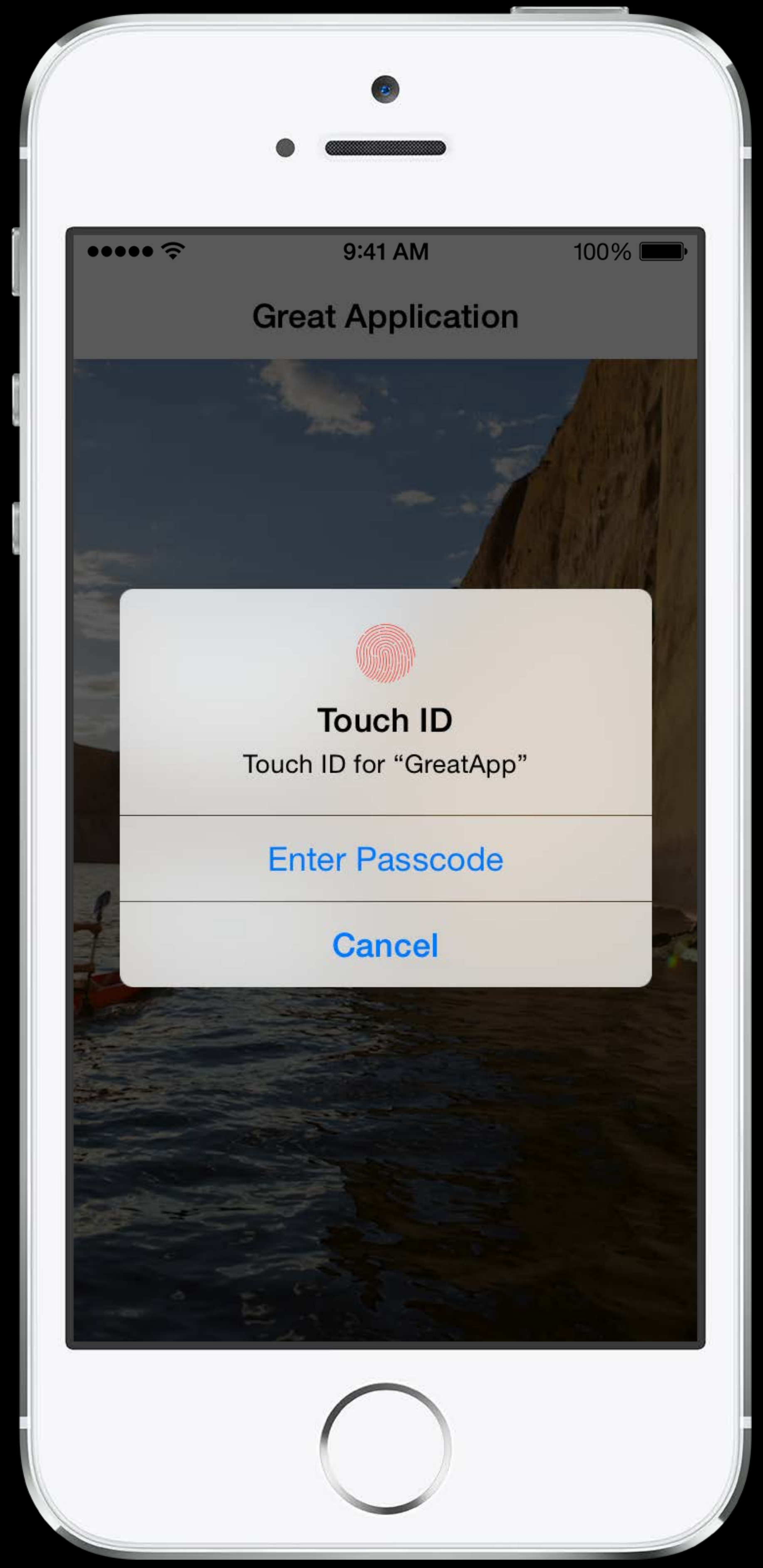


Architecture



User Interface

Standard user interface



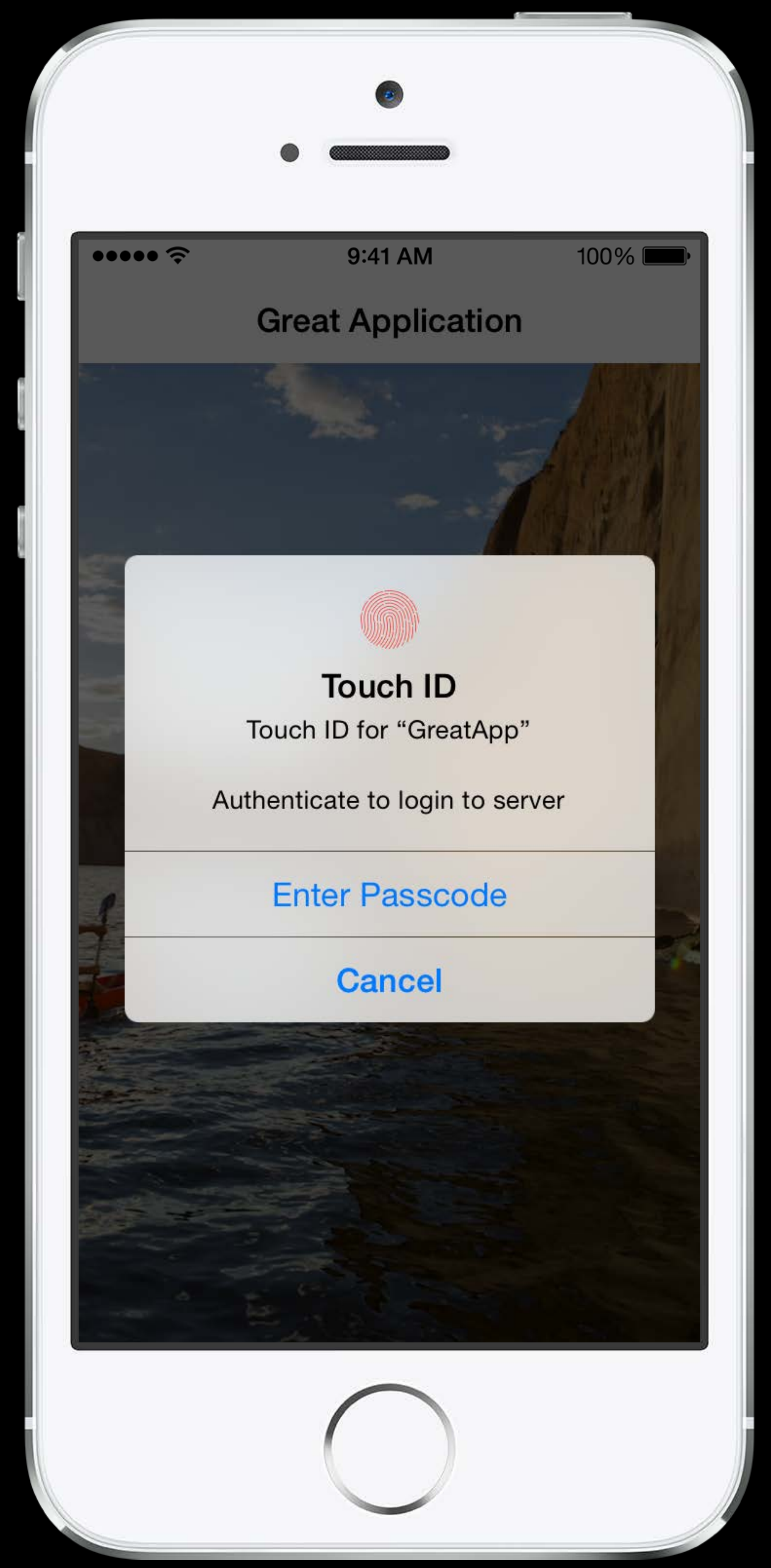
Touch ID

Touch ID for “GreatApp”

User Interface

Standard user interface

Custom prompt message recommended



Touch ID

Touch ID for “GreatApp”

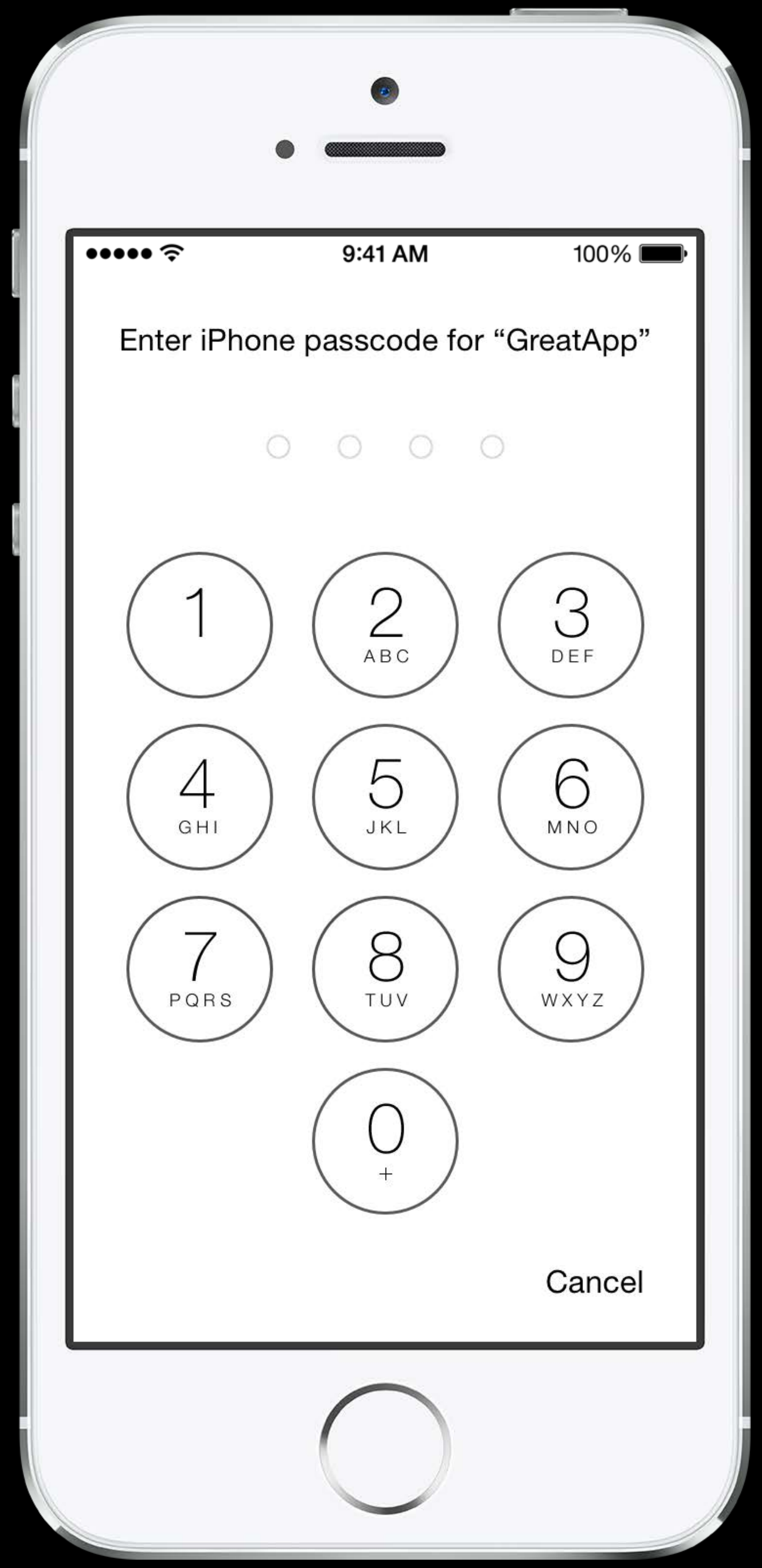
Authenticate to login to server

User Interface

Standard user interface

Custom prompt message recommended

Enter passcode option



Enter iPhone passcode for “GreatApp”

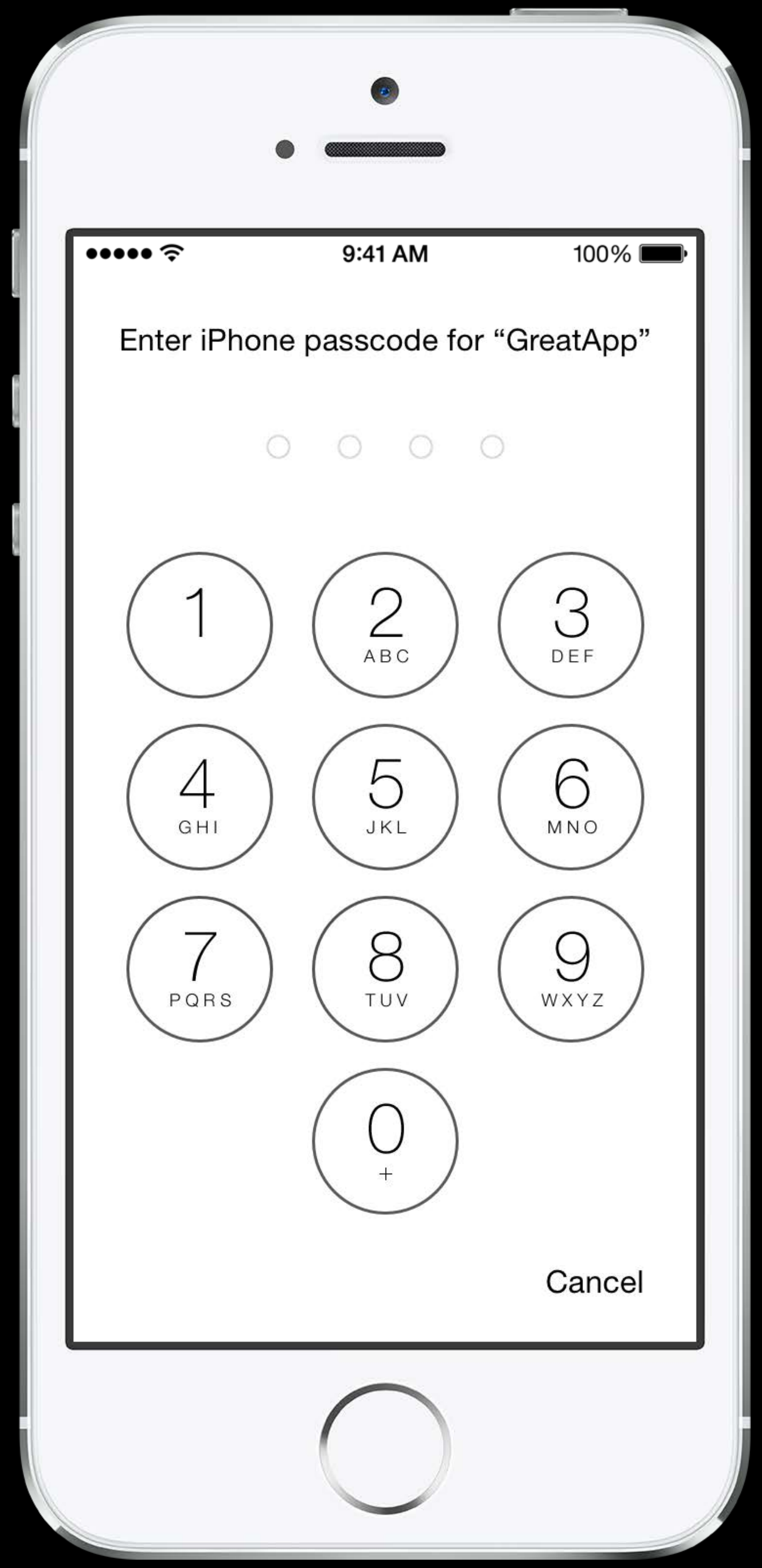
User Interface

Standard user interface

Custom prompt message recommended

Enter passcode option

Blocking keychain operation



Enter iPhone passcode for “GreatApp”

Item Accessibility with ACLs

Recommendation

Item Accessibility with ACLs

Recommendation

“When unlocked”

- `kSecAttrAccessibleWhenUnlocked`
- No passcode = item is not accessible

Item Accessibility with ACLs

Recommendation

"When unlocked"

- `kSecAttrAccessibleWhenUnlocked`
- No passcode = item is not accessible

"When unlocked and passcode set"

- `kSecAttrAccessibleWhenPasscodeSetThisDeviceOnly`
- No passcode = item is deleted

Storing a Secret

```
NSData* secret = [@"top secret" dataWithEncoding:NSUTF8StringEncoding];
NSDictionary *query = @{
    (id)kSecClass: (id)kSecClassGenericPassword,
    (id)kSecAttrService: @"myservice",
    (id)kSecAttrAccount: @"account name here",
    (id)kSecValueData: secret};


OSStatus status =  SecItemAdd((CFDictionaryRef)query, nil);
```

Storing a Secret

```
SecAccessControlRef sacObject =
SecAccessControlCreateWithFlags(kCFAllocatorDefault,
   kSecAttrAccessibleWhenPasscodeSetThisDeviceOnly,
   kSecAccessControlUserPresence, &error);
```

```
NSData* secret = [@"top secret" dataWithEncoding:NSUTF8StringEncoding];
NSDictionary *query = @{
   (id)kSecClass: (id)kSecClassGenericPassword,
   (id)kSecAttrService: @"myservice",
   (id)kSecAttrAccount: @"account name here",
   (id)kSecValueData: secret};


OSStatus status =  SecItemAdd((CFDictionaryRef)query, nil);
```

Storing a Secret

```
SecAccessControlRef sacObject =
SecAccessControlCreateWithFlags(kCFAllocatorDefault,
   kSecAttrAccessibleWhenPasscodeSetThisDeviceOnly,
   kSecAccessControlUserPresence, &error);

NSData* secret = [@"top secret" dataWithEncoding:NSUTF8StringEncoding];
NSDictionary *query = @{
   (id)kSecClass: (id)kSecClassGenericPassword,
   (id)kSecAttrService: @"myservice",
   (id)kSecAttrAccount: @"account name here",
   (id)kSecValueData: secret,
   (id)kSecAttrAccessControl: (id)sacObject};

OSStatus status =  SecItemAdd((CFDictionaryRef)query, nil);
```

Reading a Secret

```
NSDictionary *query = @{
    (id)kSecClass: (id)kSecClassGenericPassword,
    (id)kSecAttrService: @"myservice",
    (id)kSecAttrAccount: @"account name here",
    (id)kSecReturnData: @YES
};
```

```
    CFTypeRef dataTypeRef = NULL;
    OSStatus status = SecItemCopyMatching((CFDictionaryRef)query,
                                          &dataTypeRef);
```

Reading a Secret

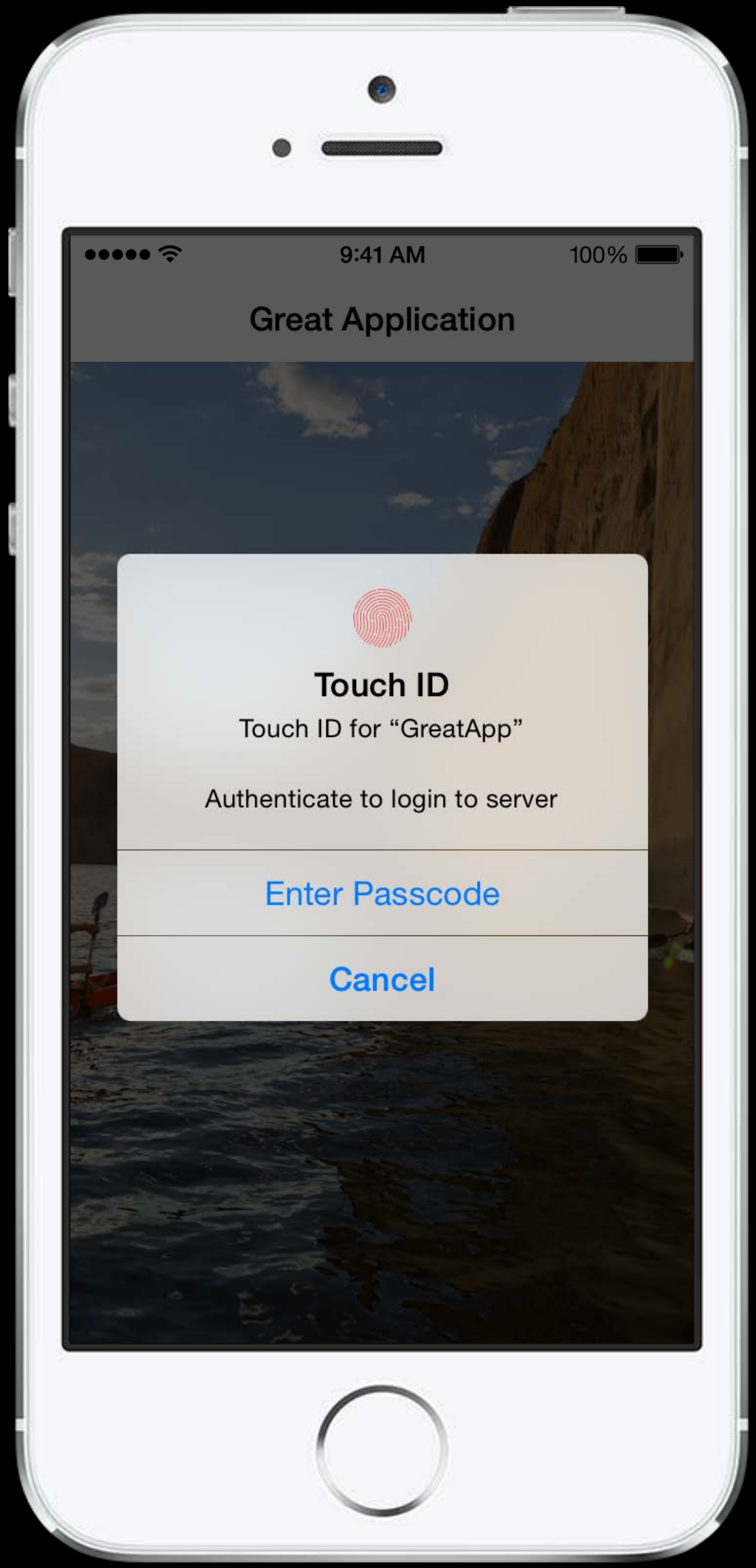
```
NSDictionary *query = @{
    (id)kSecClass: (id)kSecClassGenericPassword,
    (id)kSecAttrService: @"myservice",
    (id)kSecAttrAccount: @"account name here",
    (id)kSecReturnData: @YES,
    (id)kSecUseOperationPrompt: @"Authenticate to login to server"
};


    CFTypeRef dataTypeRef = NULL;
    OSStatus status = SecItemCopyMatching((CFDictionaryRef)query,
                                          &dataTypeRef);
```

Reading a Secret

```
NSDictionary *query = @{
    (id)kSecClass: (id)kSecClassGenericPassword,
    (id)kSecAttrService: @"myservice",
    (id)kSecAttrAccount: @"account name here",
    (id)kSecReturnData: @YES,
    (id)kSecUseOperationPrompt: @"Authenticate to login to server"
};

    CFTypeRef dataTypeRef = NULL;
    OSStatus status = SecItemCopyMatching((CFDictionaryRef)query,
                                          &dataTypeRef);
```



Touch ID

Touch ID for “GreatApp”

Authenticate to login to server

Reading a Secret

```
NSDictionary *query = @{
    (id)kSecClass: (id)kSecClassGenericPassword,
    (id)kSecAttrService: @"myservice",
    (id)kSecAttrAccount: @"account name here",
    (id)kSecReturnData: @YES,
    (id)kSecUseOperationPrompt: @"Authenticate to login to server"
};

    CFTypeRef dataTypeRef = NULL;
    OSStatus status = SecItemCopyMatching((CFDictionaryRef)query,
                                         &dataTypeRef);
```

Reading a Secret

```
NSDictionary *query = @{
    (id)kSecClass: (id)kSecClassGenericPassword,
    (id)kSecAttrService: @"myservice",
    (id)kSecAttrAccount: @"account name here",
    (id)kSecReturnData: @YES,
    (id)kSecUseOperationPrompt: @"Authenticate to login to server"
};

dispatch_async(dispatch_get_global_queue(…), ^(void){
    CFTypeRef dataTypeRef = NULL;
    OSStatus status = SecItemCopyMatching((CFDictionaryRef)query,
                                         &dataTypeRef);
});
```

Things to Consider

Things to Consider

Only foreground applications

Things to Consider

Only foreground applications

Any query may require authentication

Things to Consider

Only foreground applications

Any query may require authentication

- SecItemAdd, SecItemUpdate

Things to Consider

Only foreground applications

Any query may require authentication

- SecItemAdd, SecItemUpdate
- Broad queries—Multiple items

Things to Consider

Only foreground applications

Any query may require authentication

- SecItemAdd, SecItemUpdate
- Broad queries—Multiple items
- No authentication mode—`kSecUseNoAuthenticationUI`

Things to Consider

Only foreground applications

Any query may require authentication

- SecItemAdd, SecItemUpdate
- Broad queries—Multiple items
- No authentication mode—`kSecUseNoAuthenticationUI`

ACL protected items—No synchronization, no backup

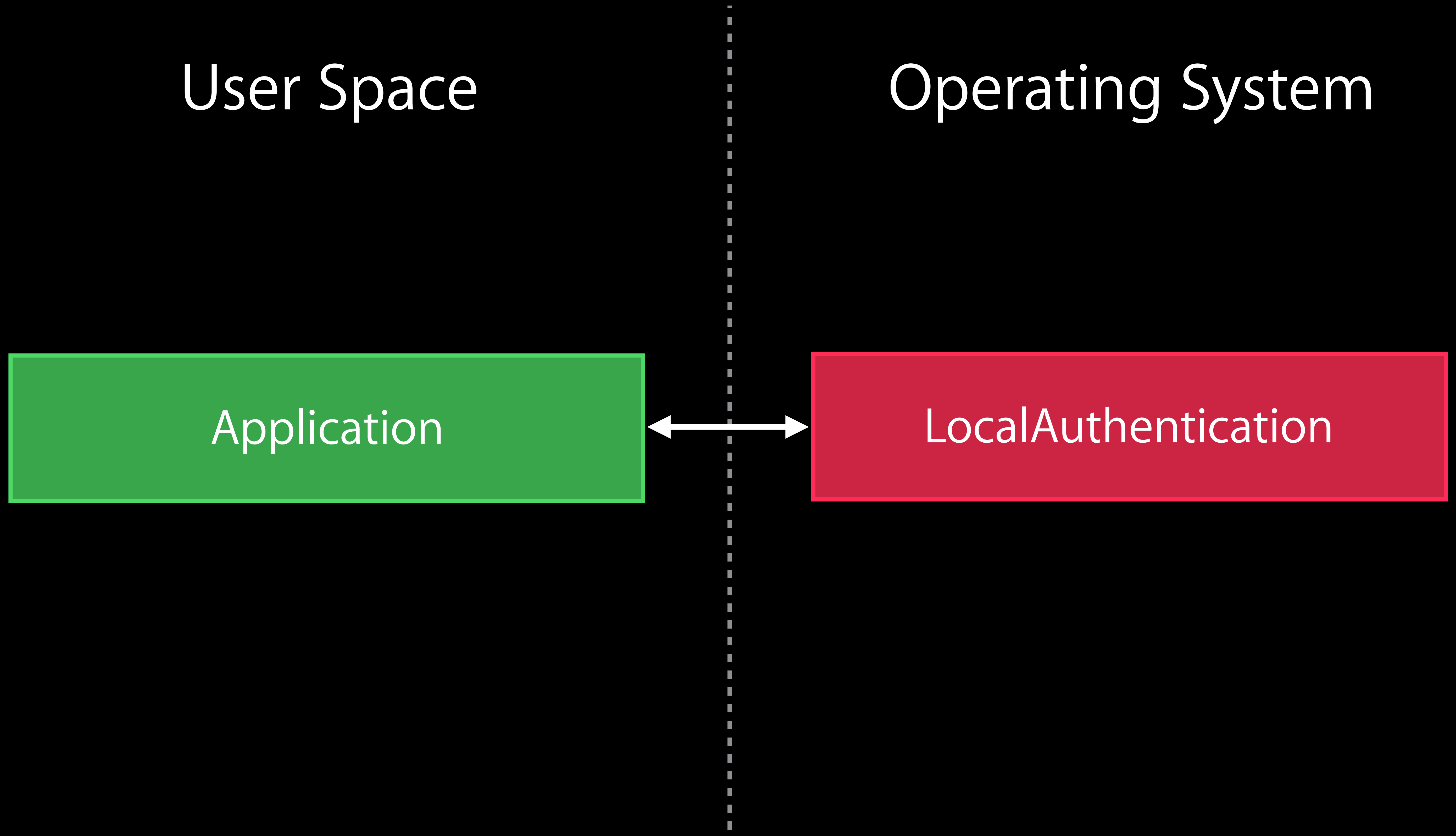
LocalAuthentication

Introduction

Credential collecting

LocalAuthentication for applications—Policy evaluation

Architecture



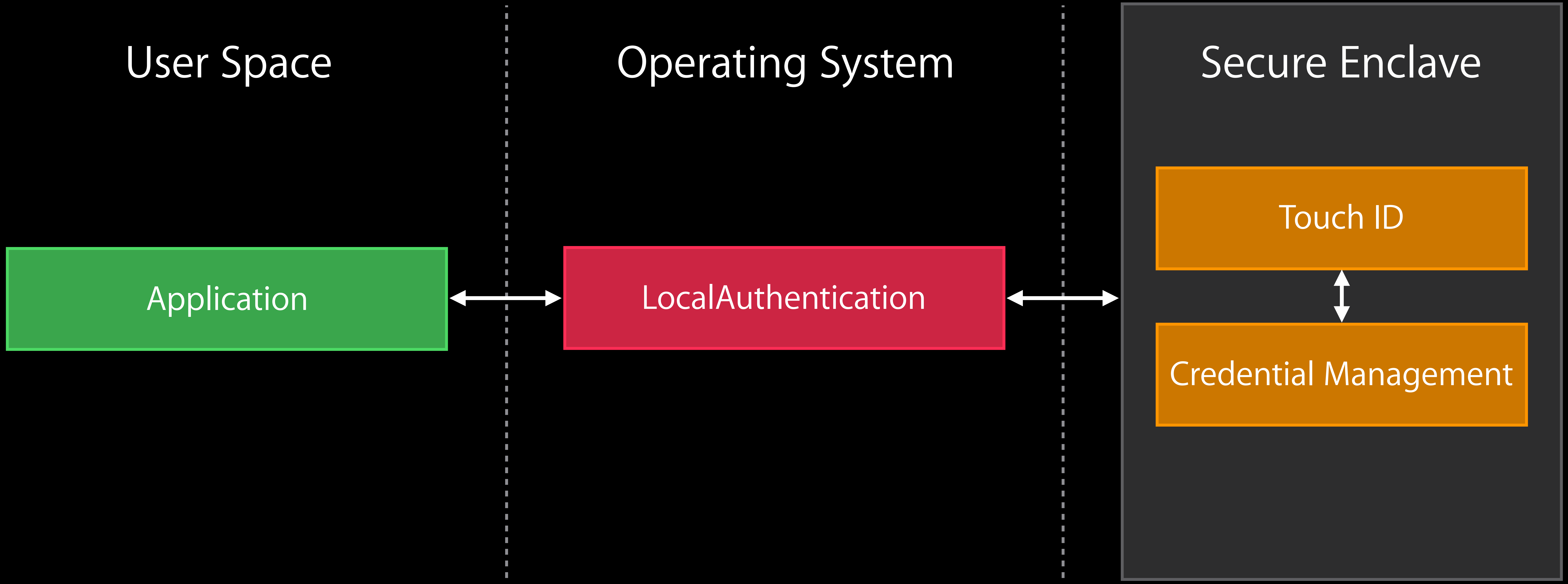
User Space

Operating System

Application

LocalAuthentication

Architecture



User Space

Operating System

Secure Enclave

Application

LocalAuthentication

Touch ID

Credential Management

LocalAuthentication Use Cases

Examples

LocalAuthentication Use Cases

Examples

Verify that user is enrolled

- Unlock features of application
- Parental check

LocalAuthentication Use Cases

Examples

Verify that user is enrolled

- Unlock features of application
- Parental check

Extension to current application's authentication

- First factor—Without passcode backup
- Second factor

LocalAuthentication Security

LocalAuthentication Security

Differs from Keychain

LocalAuthentication Security

Differs from Keychain

- Trust the OS vs. trust the secure enclave

LocalAuthentication Security

Differs from Keychain

- Trust the OS vs. trust the secure enclave

No direct access to secure enclave

LocalAuthentication Security

Differs from Keychain

- Trust the OS vs. trust the secure enclave

No direct access to secure enclave

No access to registered fingers

LocalAuthentication Security

Differs from Keychain

- Trust the OS vs. trust the secure enclave

No direct access to secure enclave

No access to registered fingers

No access to fingerprint image

LocalAuthentication API

LocalAuthentication API

Touch ID without keychain

LocalAuthentication API

Touch ID without keychain

- `canEvaluatePolicy:`

LocalAuthentication API

Touch ID without keychain

- `canEvaluatePolicy:`
- `evaluatePolicy:`

LocalAuthentication API

Touch ID without keychain

- `canEvaluatePolicy:`
- `evaluatePolicy:`
    - Yes/no answer

LocalAuthentication API

Touch ID without keychain

- `canEvaluatePolicy:`
- `evaluatePolicy:`
  - Yes/no answer
- Policy—`LAPolicyDeviceOwnerAuthenticationWithBiometrics`

LocalAuthentication API

Touch ID without keychain

- `canEvaluatePolicy:`
- `evaluatePolicy:`
  - Yes/no answer
- Policy—`LAPolicyDeviceOwnerAuthenticationWithBiometrics`
  - No passcode authentication

LocalAuthentication API

Touch ID without keychain

- `canEvaluatePolicy:`
- `evaluatePolicy:`
  - Yes/no answer
- Policy—`LAPolicyDeviceOwnerAuthenticationWithBiometrics`
  - No passcode authentication
  - Fallback to application's own password entry UI

Things to Remember

Things to Remember

Only for foreground application

Things to Remember

Only for foreground application

Policy evaluation may fail

Things to Remember

Only for foreground application

Policy evaluation may fail

Application fallback mechanism

Touch ID Available

```
//To check that policy can succeed beaded on device type and configuration

LAContext *context = [LAContext new];

NSError *error;
if([context canEvaluatePolicy:LAPolicyDeviceOwnerAuthenticationWithBiometrics
                        error:&error]) {
    NSLog(@"Touch ID is available”);
   …
}
```

Touch ID Available

```
//To check that policy can succeed beaded on device type and configuration

LAContext *context = [LAContext new];

NSError *error;
if([context canEvaluatePolicy:LAPolicyDeviceOwnerAuthenticationWithBiometrics
                        error:&error]) {
    NSLog(@"Touch ID is available”);
   …
}
```

Touch ID Authentication

```
LAContext *context = [LAContext new];
[context evaluatePolicy:LAPolicyDeviceOwnerAuthenticationWithBiometrics
        localizedReason:@"To access your photos"
                  reply:^(BOOL success, NSError *authenticationError) {
                    if (success) {
                       NSLog(@"Authenticated using Touch ID.");
                    } else {
                       NSLog(@"Authentication failed");
                    }
                 }];
```

Touch ID Authentication

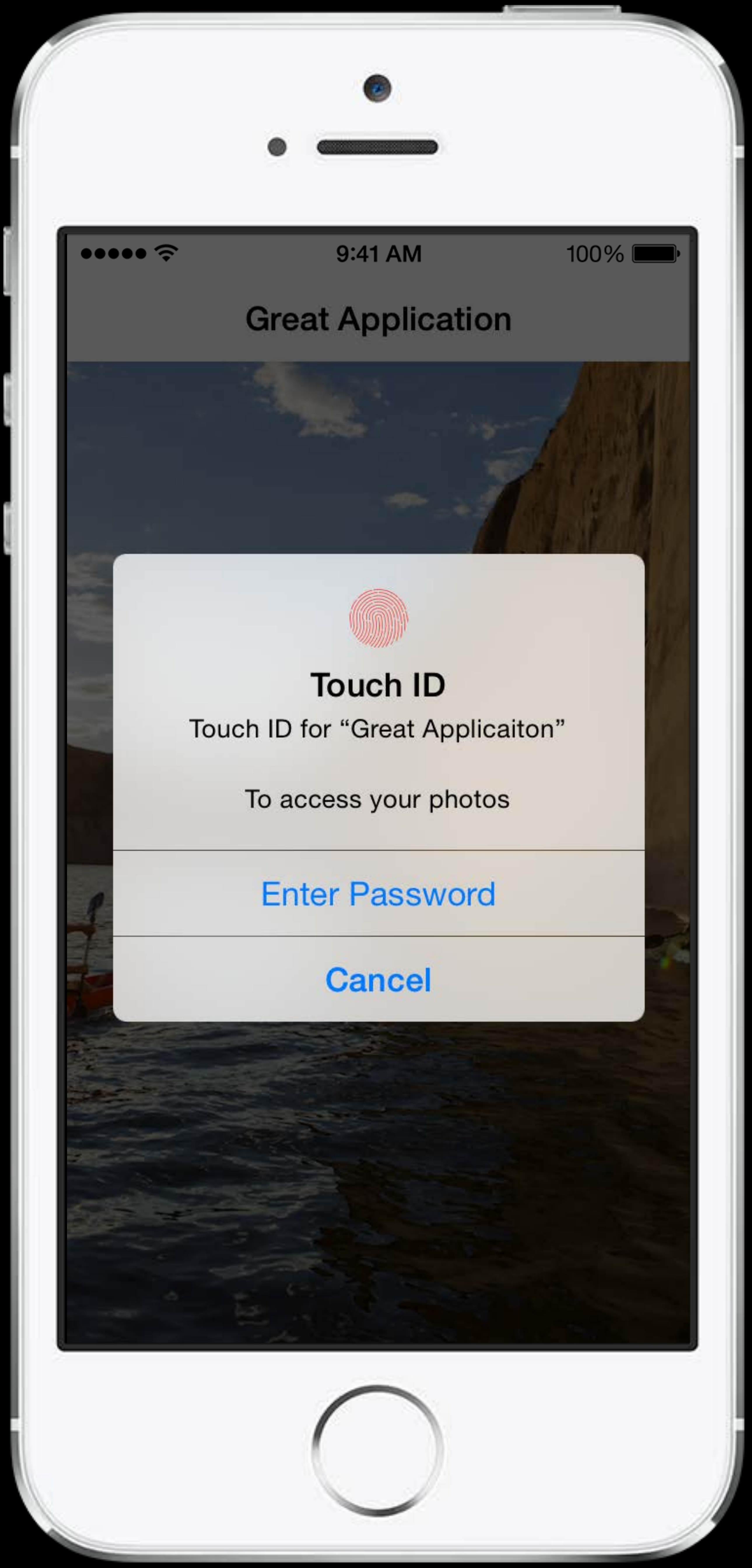
```
LAContext *context = [LAContext new];
[context evaluatePolicy:LAPolicyDeviceOwnerAuthenticationWithBiometrics
        localizedReason:@"To access your photos"
                  reply:^(BOOL success, NSError *authenticationError) {
                    if (success) {
                       NSLog(@"Authenticated using Touch ID.");
                    } else {
                       NSLog(@"Authentication failed");
                    }
                 }];
```

Touch ID Authentication

```
LAContext *context = [LAContext new];
[context evaluatePolicy:LAPolicyDeviceOwnerAuthenticationWithBiometrics
        localizedReason:@"To access your photos"
                  reply:^(BOOL success, NSError *authenticationError) {
                    if (success) {
                       NSLog(@"Authenticated using Touch ID.");
                    } else {
                       NSLog(@"Authentication failed");
                    }
                 }];
```

Touch ID Authentication

```
LAContext *context = [LAContext new];
[context evaluatePolicy:LAPolicyDeviceOwnerAuthenticationWithBiometrics
        localizedReason:@"To access your photos"
                  reply:^(BOOL success, NSError *authenticationError) {
                    if (success) {
                       NSLog(@"Authenticated using Touch ID.");
                    } else {
                       NSLog(@"Authentication failed");
                    }
                }];
```



Touch ID

Touch ID for “Great Applicaiton”

To access your photos

Touch ID Authentication

```
[context evaluatePolicy:LAPolicyDeviceOwnerAuthenticationWithBiometrics
        localizedReason:@"To access your photos"
                  reply:^(BOOL success, NSError *authenticationError) {
                    if (success) {
                       NSLog(@"Authenticated using Touch ID.");
                    } else {
                       NSLog(@"Authentication failed");
                    }
                 }];
```

Touch ID Authentication

```
[context evaluatePolicy:LAPolicyDeviceOwnerAuthenticationWithBiometrics
        localizedReason:@"To access your photos"
                  reply:^(BOOL success, NSError *error) {
                    if (success) {
                       NSLog(@"Authenticated using Touch ID.");
                    } else {
                       if (error.code == kLAErrorUserFallback) {
                          [self passcodeFallback];
                       } else if (error.code == kLAErrorUserCancel) {
                          [self cancelLogin];
                       } else {
                          [self reportError];
                       }
                    }
                 }];
```

Touch ID Authentication

```
[context evaluatePolicy:LAPolicyDeviceOwnerAuthenticationWithBiometrics
        localizedReason:@"To access your photos"
                  reply:^(BOOL success, NSError *error) {
                    if (success) {
                       NSLog(@"Authenticated using Touch ID.");
                    } else {
                       if (error.code == kLAErrorUserFallback) {
                           [self passcodeFallback];
                       } else if (error.code == kLAErrorUserCancel) {
                           [self cancelLogin];
                       } else {
                           [self reportError];
                       }
                    }
                 }];
```

Touch ID Authentication

```
[context evaluatePolicy:LAPolicyDeviceOwnerAuthenticationWithBiometrics
        localizedReason:@"To access your photos"
                  reply:^(BOOL success, NSError *error) {
                    if (success) {
                        NSLog(@"Authenticated using Touch ID.");
                    } else {
                        if (error.code == kLAErrorUserFallback) {
                           [self passcodeFallback];
                        } else if (error.code == kLAErrorUserCancel) {
                           [self cancelLogin];
                        } else {
                           [self reportError];
                        }
                    }
                 }];
```

Touch ID Authentication

```
[context evaluatePolicy:LAPolicyDeviceOwnerAuthenticationWithBiometrics
        localizedReason:@"To access your photos"
                  reply:^(BOOL success, NSError *error) {
                    if (success) {
                       NSLog(@"Authenticated using Touch ID.");
                    } else {
                       if (error.code == kLAErrorUserFallback) {
                           [self passcodeFallback];
                       } else if (error.code == kLAErrorUserCancel) {
                           [self cancelLogin];
                       } else {
                           [self reportError];
                       }
                    }
                 }];
```

Touch ID Authentication

```
[context evaluatePolicy:LAPolicyDeviceOwnerAuthenticationWithBiometrics
        localizedReason:@"To access your photos"
                  reply:^(BOOL success, NSError *error) {
                    if (success) {
                       NSLog(@"Authenticated using Touch ID.");
                    } else {
                       if (error.code == kLAErrorUserFallback) {
                           [self passcodeFallback];
                       } else if (error.code == kLAErrorUserCancel) {
                           [self cancelLogin];
                       } else {
                           [self reportError];
                       }
                    }
                 }];
```

Demo

Summary

Summary

Keychain is here for your secrets

Summary

Keychain is here for your secrets

What is new?

Summary

Keychain is here for your secrets

What is new?

- New accessibility class

Summary

Keychain is here for your secrets

What is new?

- New accessibility class
  - Items invalidated with no passcode

Summary

Keychain is here for your secrets

What is new?

- New accessibility class
  - Items invalidated with no passcode
- Keychain item ACLs

Summary

Keychain is here for your secrets

What is new?

- New accessibility class
  - Items invalidated with no passcode
- Keychain item ACLs
  - Protection using Touch ID or passcode

Summary

Keychain is here for your secrets

What is new?

- New accessibility class
  - Items invalidated with no passcode
- Keychain item ACLs
  - Protection using Touch ID or passcode
- LocalAuthentication

Summary

Keychain is here for your secrets

What is new?

- New accessibility class
    - Items invalidated with no passcode
- Keychain item ACLs
    - Protection using Touch ID or passcode
- LocalAuthentication
    - To start Touch ID authentication

More Information

Paul Danbold
Core OS Technologies Evangelist
danbold@apple.com

Documentation

iOS Security White Paper
http://images.apple.com/ipad/business/docs/iOS_Security_Feb14.pdf

Apple Developer Forums
http://devforums.apple.com

Related Sessions

● User Privacy in iOS and OS X	Nob Hill	Thursday 2:00PM

Labs

● Security Lab	Core OS Lab B	Wednesday 11:30AM
● Security and Privacy Lab	Core OS Lab B	Thursday 3:15PM

WWDC14