Advanced Keychain Use Cases

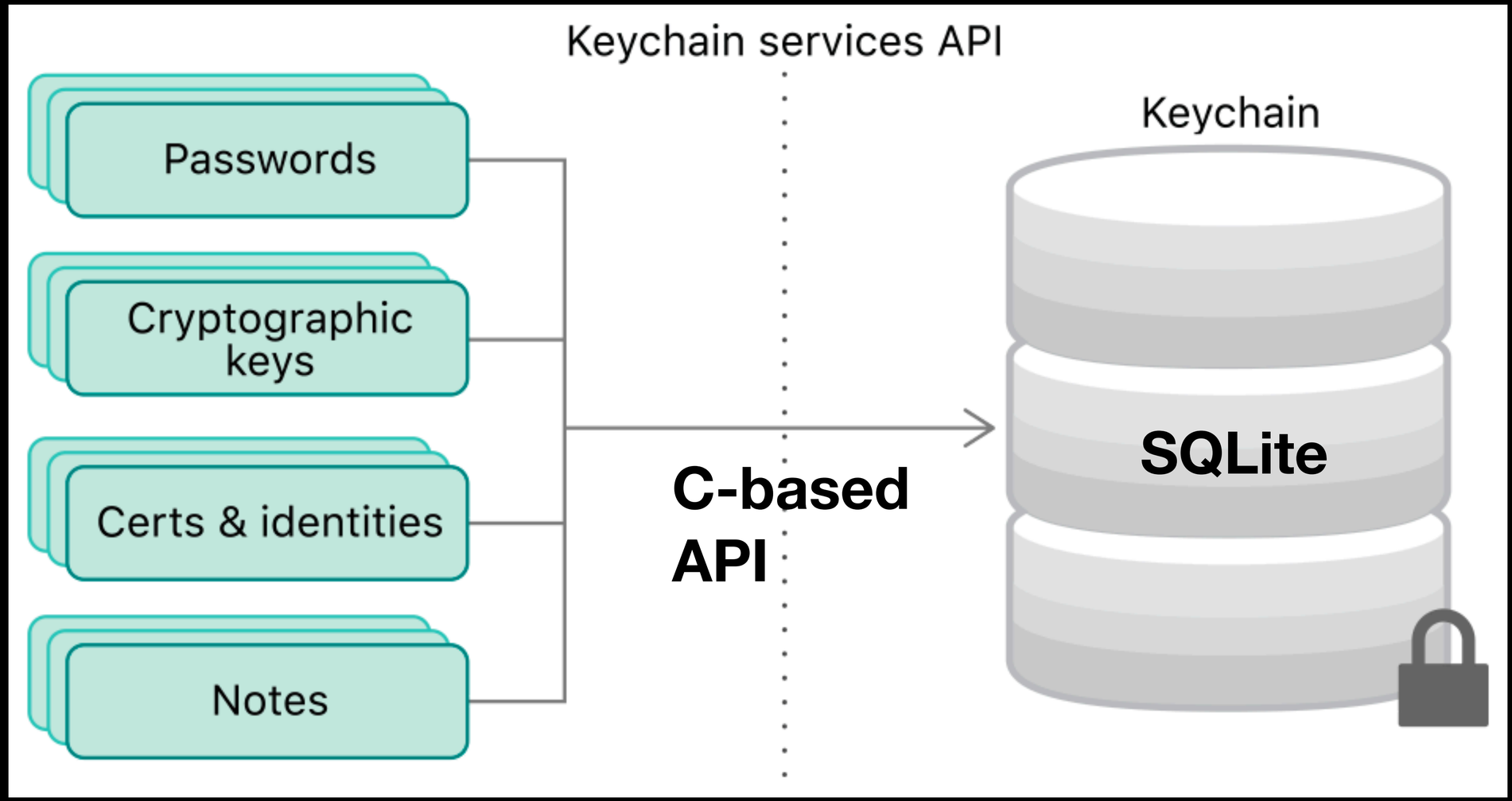
Advanced Keychain Use Cases

- Keychain Sharing
- Biometrics

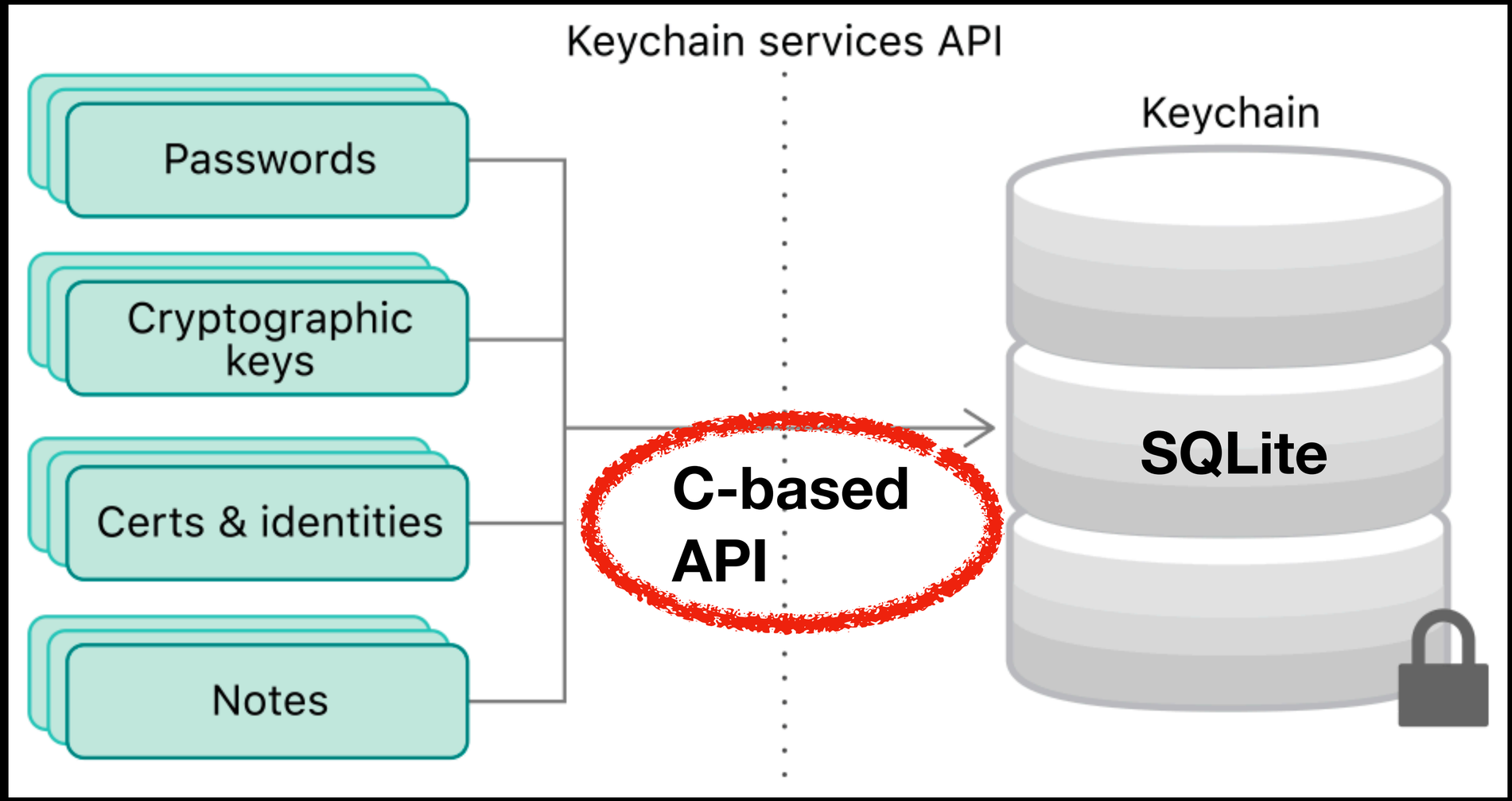
Keychain: Recap

- Storage for sensitive data
- Backed by SQLite
- C-based API
- Handles encryption and decryption in background
- Integrated with Biometrics, App Security Groups, Passcode protection

Keychain: Recap



Keychain: Recap



Keychain API

Keychain API

Sample Query

```
var query = [String: Any]()
query[kSecAttrAccount as String] = key
query[kSecReturnData as String] = kCFBooleanTrue
query[kSecMatchLimit as String] = kSecMatchLimitOne
query[kSecReturnAttributes as String] = kCFBooleanTrue
query[kSecUseAuthenticationContext as String] = context


var queryResult: CFTypeRef?
let status = SecItemCopyMatching(query as CFDictionary, &queryResult)

guard [errSecItemNotFound, errSecSuccess]
    .contains(status) else { return nil }

if let result = queryResult as? [String: Any] {
    return result[kSecValueData as String] as? Data
}
```

Keychain API

Procedural Programming Paradigm

Keychain API

Procedural Programming Paradigm

Cons

- Low-level
- No/little encapsulation
- No predefined data types

Keychain API

Procedural Programming Paradigm

Cons

- Low-level
- No/little encapsulation
- No predefined data types

Pros

- Very flexible
- Bindings can be created to other paradigms
- Can be tailored to a particular domain

Keychain Wrappers

743 results (339 ms)

Sort by: Most stars | Save | ...

kishikawakatsumi/KeychainAccess

Unstar | Sponsor

Simple Swift wrapper for **Keychain** that works on iOS, watchOS, tvOS and macOS.

security keychain touch-id

Swift · ☆ 7.4k · Updated 24 days ago

square/Valet

Unstar

Valet lets you securely store data in the iOS, tvOS, or macOS **Keychain** without knowing a thing about how the **Keychain** works. It's easy. W...

macos security ios crypto tvos

Swift · ☆ 3.9k · Updated 2 days ago

matthewpalmer/Locksmith

Star

A powerful, protocol-oriented library for working with the **keychain** in Swift.

Swift · ☆ 2.9k · Updated on Nov 21, 2022

matthewpalmer/Locksmith

A powerful, protocol-oriented library for working with the **keychain** in S

Swift · ☆ 2.9k · Updated on Nov 21, 2022

evgenyneu/keychain-swift

Helper functions for saving text in **Keychain** securely for iOS, OS X, tvO watchOS.

swift ios keychain

Swift · ☆ 2.4k · Updated 7 hours ago

jrendel/SwiftKeychainWrapper Public archive

A simple wrapper for the iOS **Keychain** to allow you to use it in a similar User Defaults. Written in Swift.

Swift · ☆ 1.6k · Updated on Jan 10

Building a Keychain Wrapper

Keychain API

Sample Query

```
var query = [String: Any]()
query[kSecAttrAccount as String] = key
query[kSecReturnData as String] = kCFBooleanTrue
query[kSecMatchLimit as String] = kSecMatchLimitOne
query[kSecReturnAttributes as String] = kCFBooleanTrue
query[kSecUseAuthenticationContext as String] = context
```

1. Query configuration

```
var queryResult: CFTypeRef?
let status = SecItemCopyMatching(query as CFDictionary, &queryResult)

guard [errSecItemNotFound, errSecSuccess]
    .contains(status) else { return nil }

if let result = queryResult as? [String: Any] {
    return result[kSecValueData as String] as? Data
}
```

Keychain API

Sample Query

```
var query = [String: Any]()
query[kSecAttrAccount as String] = key
query[kSecReturnData as String] = kCFBooleanTrue
query[kSecMatchLimit as String] = kSecMatchLimitOne
query[kSecReturnAttributes as String] = kCFBooleanTrue
query[kSecUseAuthenticationContext as String] = context
```

1. Query configuration

```
var queryResult: CFTypeRef?
let status = SecItemCopyMatching(query as CFDictionary, &queryResult)

guard [errSecItemNotFound, errSecSuccess]
    .contains(status) else { return nil }

if let result = queryResult as? [String: Any] {
    return result[kSecValueData as String] as? Data
}
```

2. Query execution

Keychain API

Sample Query

```
var query = [String: Any]()
query[kSecAttrAccount as String] = key
query[kSecReturnData as String] = kCFBooleanTrue
query[kSecMatchLimit as String] = kSecMatchLimitOne
query[kSecReturnAttributes as String] = kCFBooleanTrue
query[kSecUseAuthenticationContext as String] = context
```

1. Query configuration

```
var queryResult: CFTypeRef?
let status = SecItemCopyMatching(query as CFDictionary, &queryResult)
```

2. Query execution

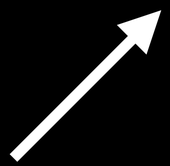
```
guard [errSecItemNotFound, errSecSuccess]
    .contains(status) else { return nil }

if let result = queryResult as? [String: Any] {
    return result[kSecValueData as String] as? Data
}
```

3. Result handling

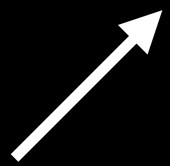
Keychain API

Query Execution

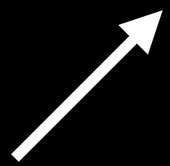
```
var queryResult: CFTypeRef?
let status = SecItemCopyMatching(query as CFDictionary, &queryResult)
```



Error Code (Int32)



Query/Input



Output

OSStatus.com

Look up Apple API errors quickly!

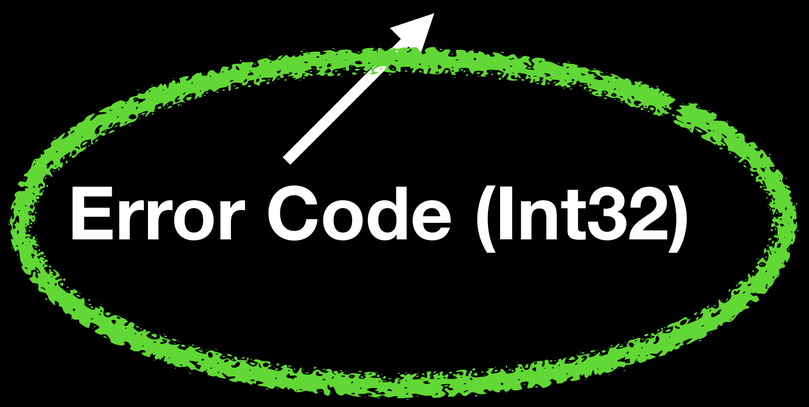
25297

Platform	Framework	Error Name	Error Code	Description
	CarbonCore MacErrors.h	errKCDuplicateCallback	-25297	

Keychain API

Query Execution

```
var queryResult: CFTypeRef?
let status = SecItemCopyMatching(query as CFDictionary, &queryResult)
```



Error Code (Int32)

Query/Input

Output

Let’s fix this!

Keychain API

Fixing Errors

```
func makeNSError(osStatus: OSStatus) -> Error {
    let description = SecCopyErrorMessageString(osStatus, nil)
    return NSError(domain: NSOSStatusErrorDomain,
                   code: Int(osStatus),
                   userInfo: [NSLocalizedDescriptionKey: description as Any])
}
```

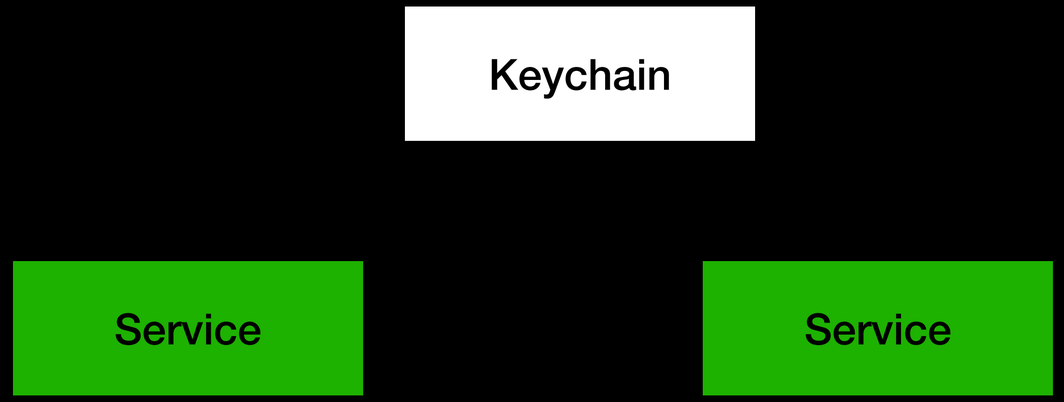
Storing Data

Keychain API

Storing Data

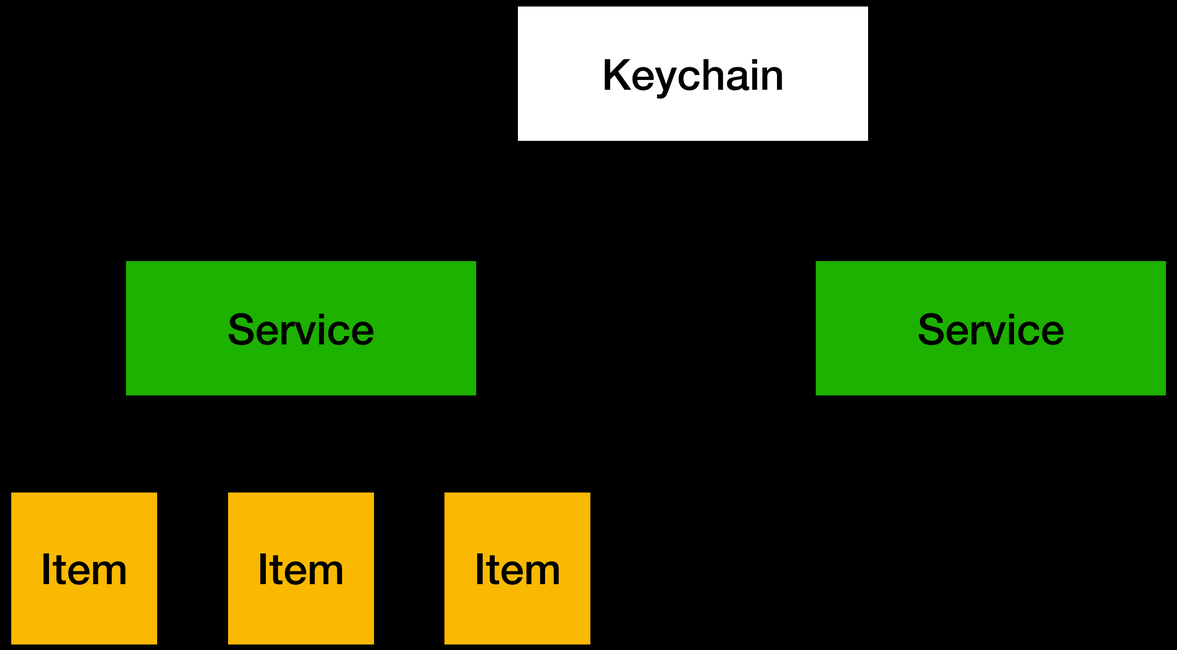
Keychain API

Storing Data



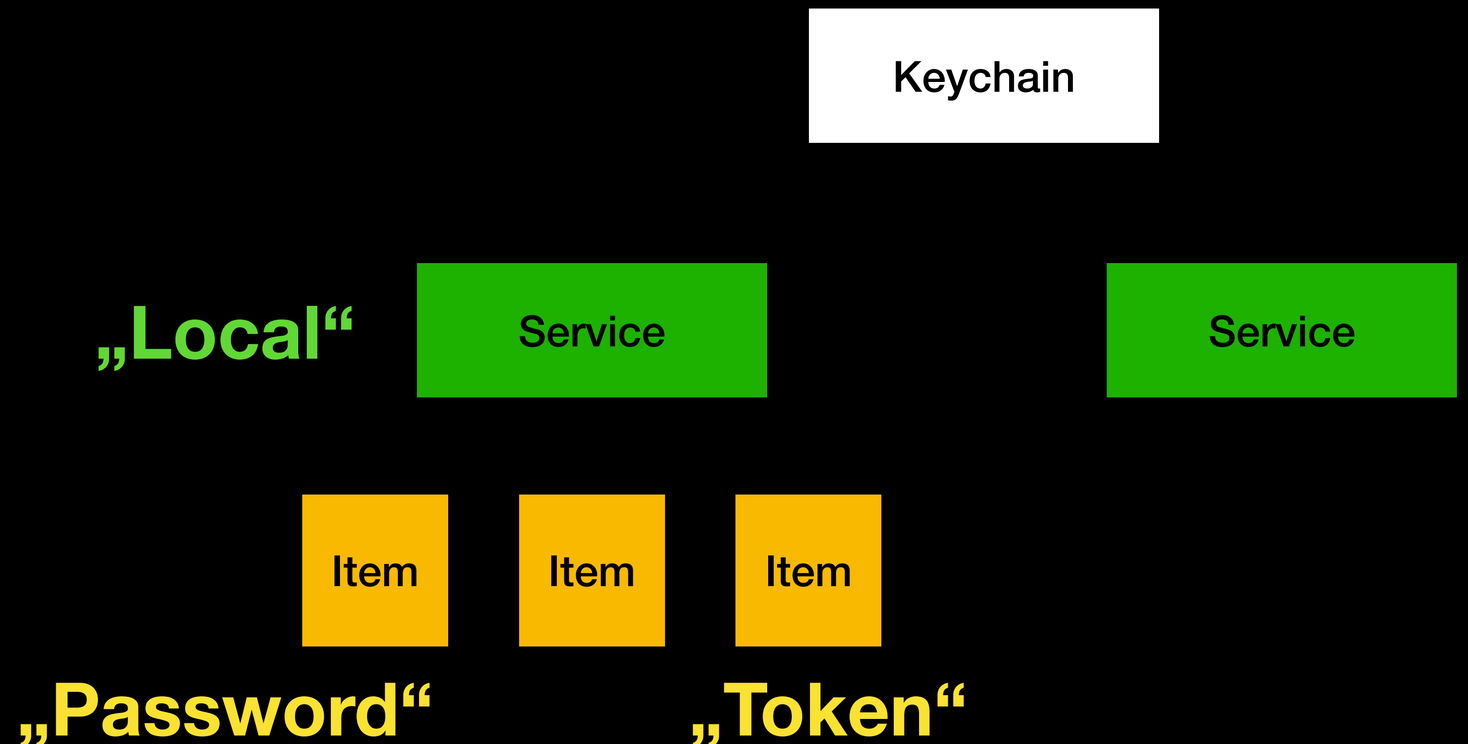
Keychain API

Storing Data



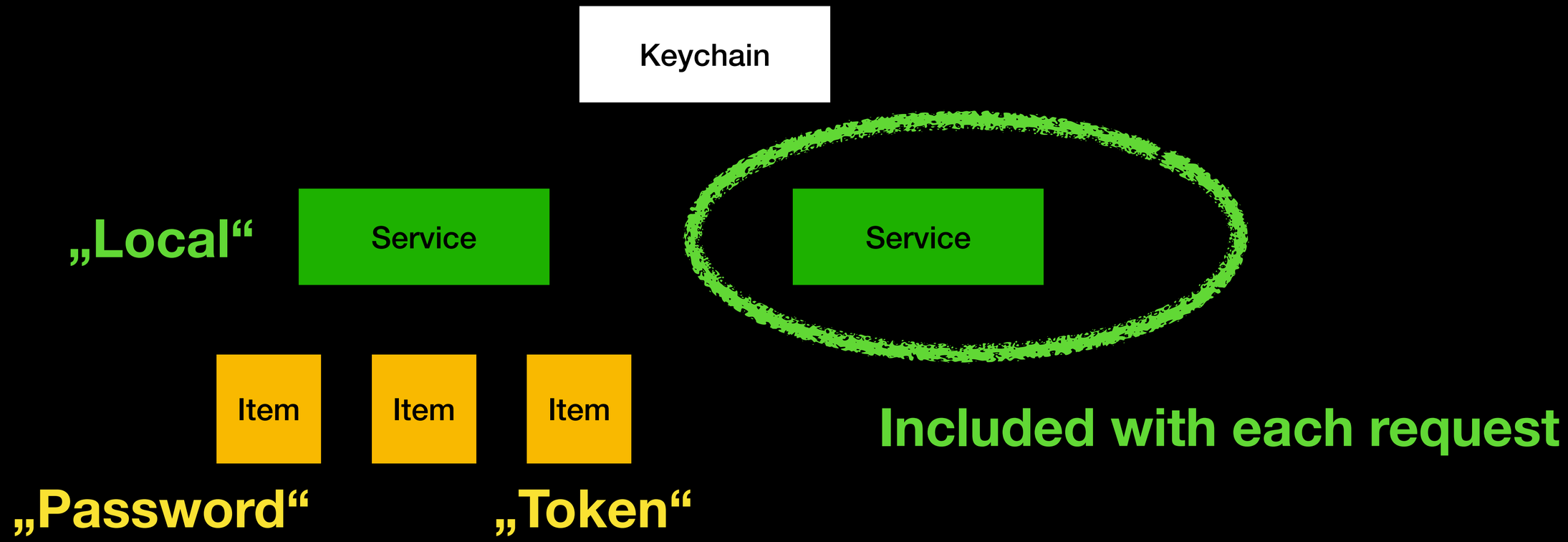
Keychain API

Storing Data



Keychain API

Storing Data



Let’s Make a Class

Keychain API

Storing Data

```
public final class Keychain {
    private let name: String?

    public init(name: String?) {
        self.name = name
    }
}
```

Keychain API

Storing Data: Building a query

```
private var baseQuery: [String: Any] {
    var query = [String: Any]()
    query[kSecClass as String] = kSecClassGenericPassword
    if let name = name {
        query[kSecAttrService as String] = name
    }
    return query
}
```

Keychain API

Storing Data: Building a query

```
private var baseQuery: [String: Any] {
    var query = [String: Any]()
    query[kSecClass as String] = kSecClassGenericPassword
    if let name = name {
        query[kSecAttrService as String] = name
    }
    return query
}
```

Keychain API

Removing Data

```
private func remove(key: String) throws {
    var query = baseQuery
    query[kSecAttrAccount as String] = key

    let result = SecItemDelete(query as CFDictionary)
    if result != errSecItemNotFound && result != errSecSuccess {
        let error = makeNSError(osStatus: result)
        throw error
    }
}
```

Keychain API

Removing Data

```
private func remove(key: String) throws {
    var query = baseQuery
    query[kSecAttrAccount as String] = key

    let result = SecItemDelete(query as CFDictionary)
    if result != errSecItemNotFound && result != errSecSuccess {
        let error = makeNSError(osStatus: result)
        throw error
    }
}
```

Keychain API

Setting Data

```
public func set(key: String, value: Data) throws {
    try remove(key)

    var query = baseQuery
    query[kSecAttrAccount as String] = key
    query[kSecValueData as String] = value

    let result = SecItemAdd(query as CFDictionary, nil)

    if result != errSecSuccess {
        let error = makeNSError(osStatus: result)
        throw error
    }
}
```

Keychain API

Setting Data

```
public func set(key: String, value: Data) throws {
    try remove(key)

    var query = baseQuery
    query[kSecAttrAccount as String] = key
    query[kSecValueData as String] = value

    let result = SecItemAdd(query as CFDictionary, nil)

    if result != errSecSuccess {
        let error = makeNSError(osStatus: result)
        throw error
    }
}
```

Keychain API

Getting Data

```
public func get(key: String) throws -> Data? {
    var query = baseQuery
    query[kSecAttrAccount as String] = key
    query[kSecReturnData as String] = kCFBooleanTrue
    query[kSecMatchLimit as String] = kSecMatchLimitOne
    query[kSecReturnAttributes as String] = kCFBooleanTrue

    var queryResult: CFTypeRef?
    let status = SecItemCopyMatching(query as CFDictionary, &queryResult)

    guard [errSecItemNotFound, errSecSuccess]
        .contains(status) else {
        let error = makeNSError(osStatus: status)
        throw error
    }

    if let result = queryResult as? [String: Any] {
        return result[kSecValueData as String] as? Data
    }
    return nil
}
```

Keychain API

Getting Data

```
public func get(key: String) throws -> Data? {
    var query = baseQuery
    query[kSecAttrAccount as String] = key
    query[kSecReturnData as String] = kCFBooleanTrue
    query[kSecMatchLimit as String] = kSecMatchLimitOne
    query[kSecReturnAttributes as String] = kCFBooleanTrue

    var queryResult: CFTypeRef?
    let status = SecItemCopyMatching(query as CFDictionary, &queryResult)

    guard [errSecItemNotFound, errSecSuccess]
        .contains(status) else {
        let error = makeNSError(osStatus: status)
        throw error
    }

    if let result = queryResult as? [String: Any] {
        return result[kSecValueData as String] as? Data
    }
    return nil
}
```

Keychain API

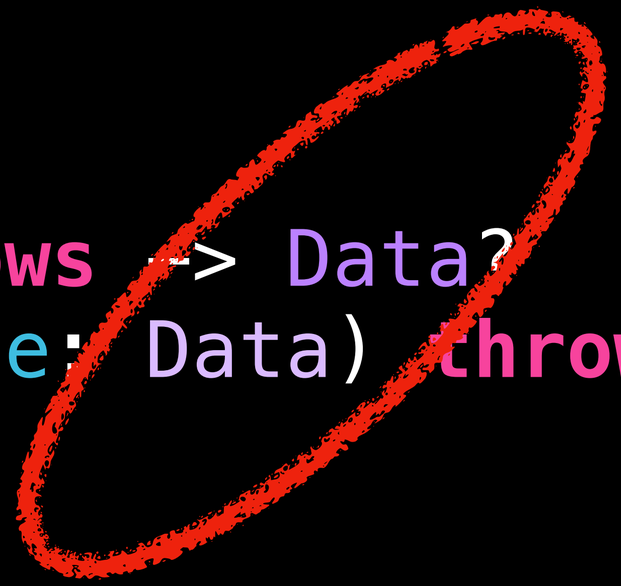
Other Data Types

```
public func get(key: String) throws -> Data?
public func set(key: String, value: Data) throws
```

Keychain API

Other Data Types

```
public func get(key: String) throws -> Data?
public func set(key: String, value: Data) throws
```



Keychain API

Other Data Types

```
public func get(key: String) throws -> String? {
    let data: Data? = try get(key: key)
    guard let data else {
        return nil
    }
    return String(data: data, encoding: .utf8)
}
```

Keychain API

Other Data Types: String

```
public func get(key: String) throws -> String? {
    let data: Data? = try get(key: key)
    guard let data else {
        return nil
    }
    return String(data: data, encoding: .utf8)
}
```

Keychain API

Other Data Types: NSNumbers

```
public func get(key: String) throws -> Bool? {
    guard let string: String = try get(key: key) else {
        return nil
    }
    return (string as NSString).boolValue
}

public func get(key: String) throws -> Int? {
    guard let string: String = try get(key: key) else {
        return nil
    }
    return (string as NSString).integerValue
}

public func get(key: String) throws -> Double? {
    guard let string: String = try get(key: key) else {
        return nil
    }
    return (string as NSString).doubleValue
}
```

Keychain API

Other Data Types: NSNumbers

```
public func get(key: String) throws -> Bool? {
    guard let string: String = try get(key: key) else {
        return nil
    }
    return (string as NSString).boolValue
}

public func get(key: String) throws -> Int? {
    guard let string: String = try get(key: key) else {
        return nil
    }
    return (string as NSString).integerValue
}

public func get(key: String) throws -> Double? {
    guard let string: String = try get(key: key) else {
        return nil
    }
    return (string as NSString).doubleValue
}
```

Biometrics

Biometrics
How do you detect a change in biometric state?

Biometrics

Detect Biometric State Change

```
let context = LAContext()
context.canEvaluatePolicy(.DeviceOwnerAuthenticationWithBiometrics, error: nil)

if let domainState = context.evaluatedPolicyDomainState
    where domainState == oldDomainState  {
    // Enrollment state the same

} else {
    // Enrollment state changed

}
```

Biometrics

Detect Biometric State Change

e1Aj1oan

Biometrics

Detect Biometric State Change

e1Aj1oan

Gfn12km

Biometrics

Detect Biometric State Change

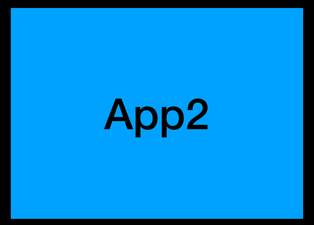
e1Aj1oan

Gfn12km

Biometric State Changed!

Biometrics

Detect Biometric State Change



e1Aj1oan

Gfn12km

fdk3fbh

3f9keB

Not matching across multiple apps…

Biometrics

Detect Biometric State Change

e1Aj1oan

Gfn12km

Gd39fk1

K239gna

fdk3fbh

3f9keB

… and even the same app and it's App Extension!

Biometry-Protected Keychain

Biometry-Protected Keychain

- No access to data if the user not authenticated
- Values are erased when biometric state changes

Biometry-Protected Keychain

Add an attribute…

```
private func appendAccessControlAttributes(to query: inout [String: Any],
                                           accessControlLevel: AccessControlLevel) {
    if accessControlLevel == .biometric {
        query[kSecAttrAccessControl as String] = SecAccessControlCreateWithFlags(kCFAllocatorDefault,
                                                                                 kSecAttrAccessibleWhenUnlockedThisDeviceOnly,
                                                                                 .biometryCurrentSet,
                                                                                 nil)

    }
}
```

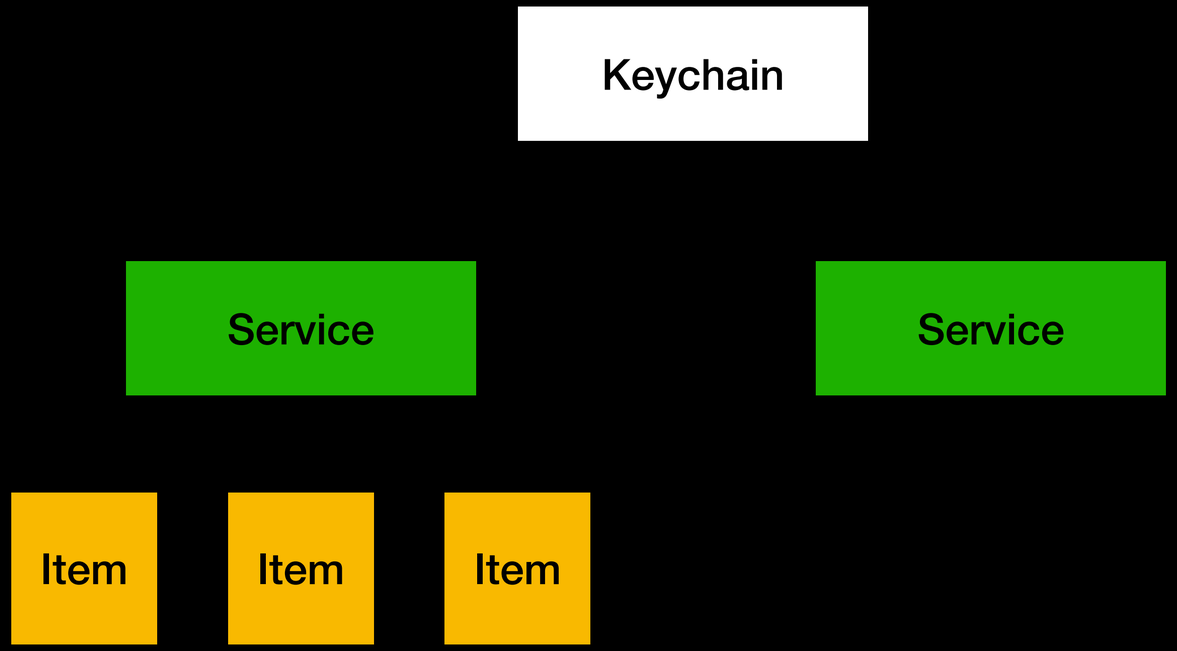
Biometry-Protected Keychain

… use in our existing methods

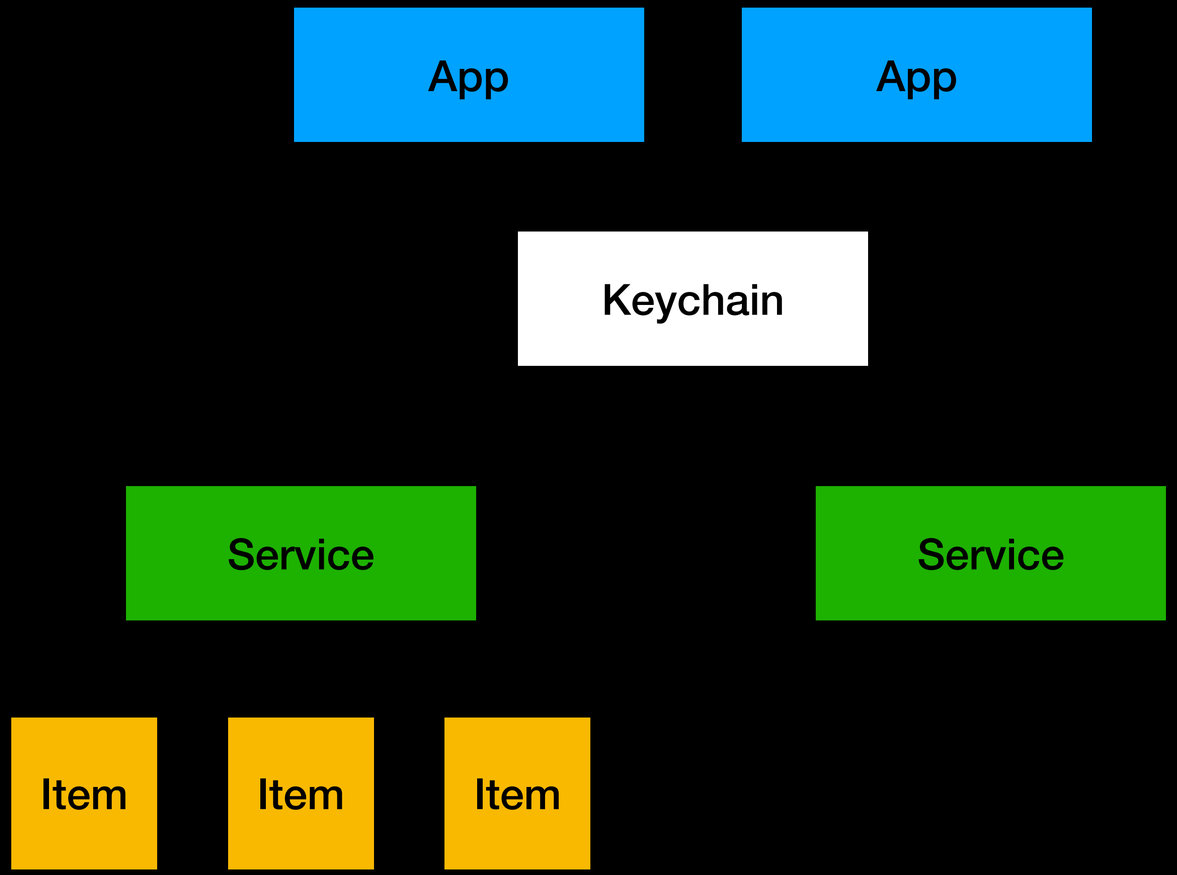
```
public func get(key: String, accessControlLevel: AccessControlLevel = .none) throws -> Data? {
    if accessControlLevel == .biometric {
        query[kSecUseAuthenticationContext as String] = context
    }

    appendAccessControlAttributes(to: &query, accessControlLevel: accessControlLevel)

    var queryResult: CFTypeRef?
    let status = SecItemCopyMatching(query as CFDictionary, &queryResult)

    if let result = queryResult as? [String: Any] {
        return result[kSecValueData as String] as? Data
    }
    return nil
}
```

Keychain Sharing

Keychain Sharing

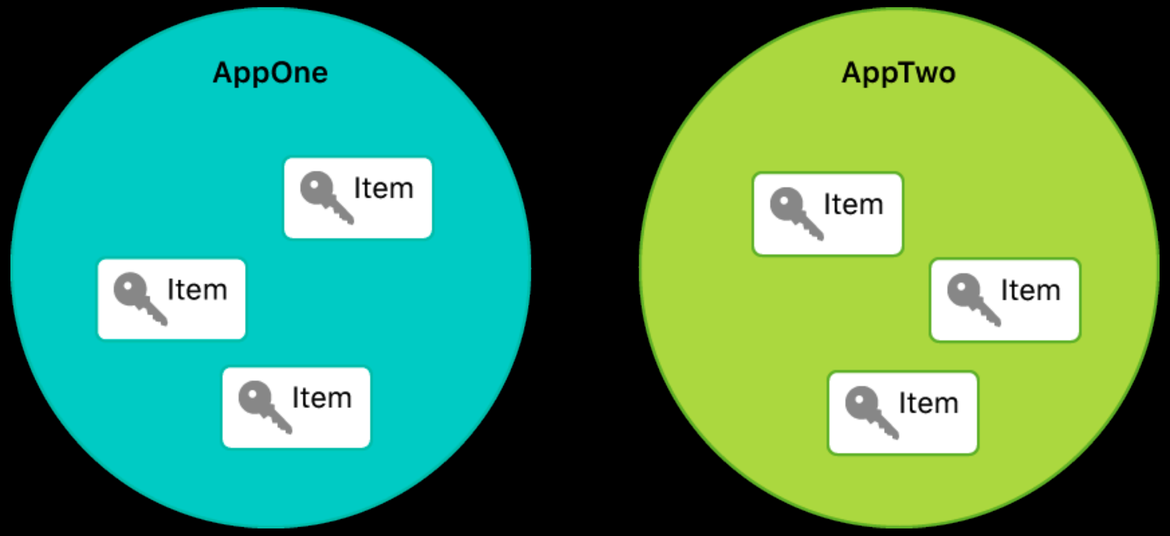


Keychain Sharing



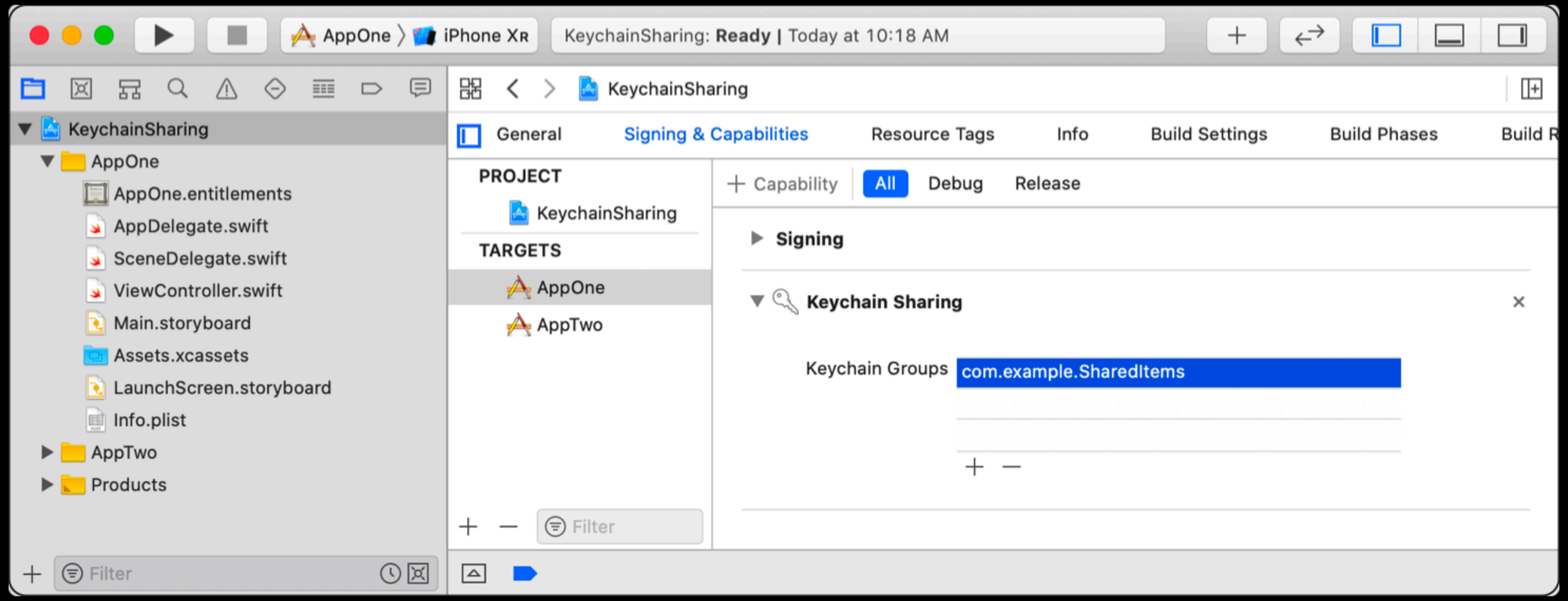
Keychain Sharing

No sharing



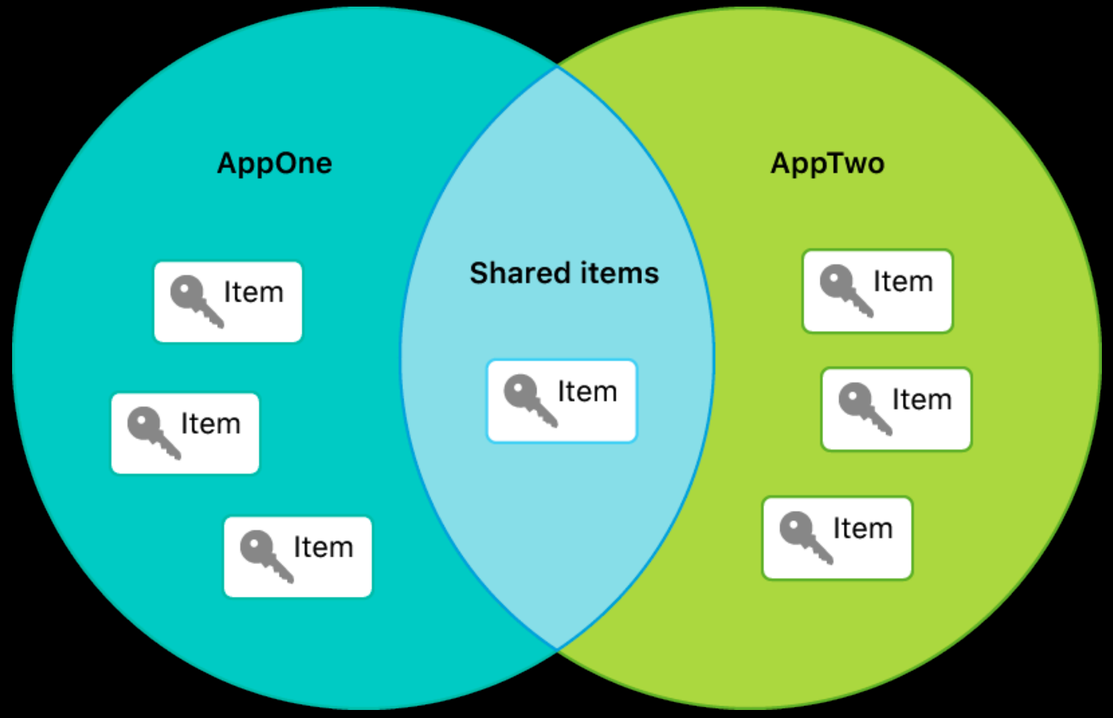
Keychain Sharing

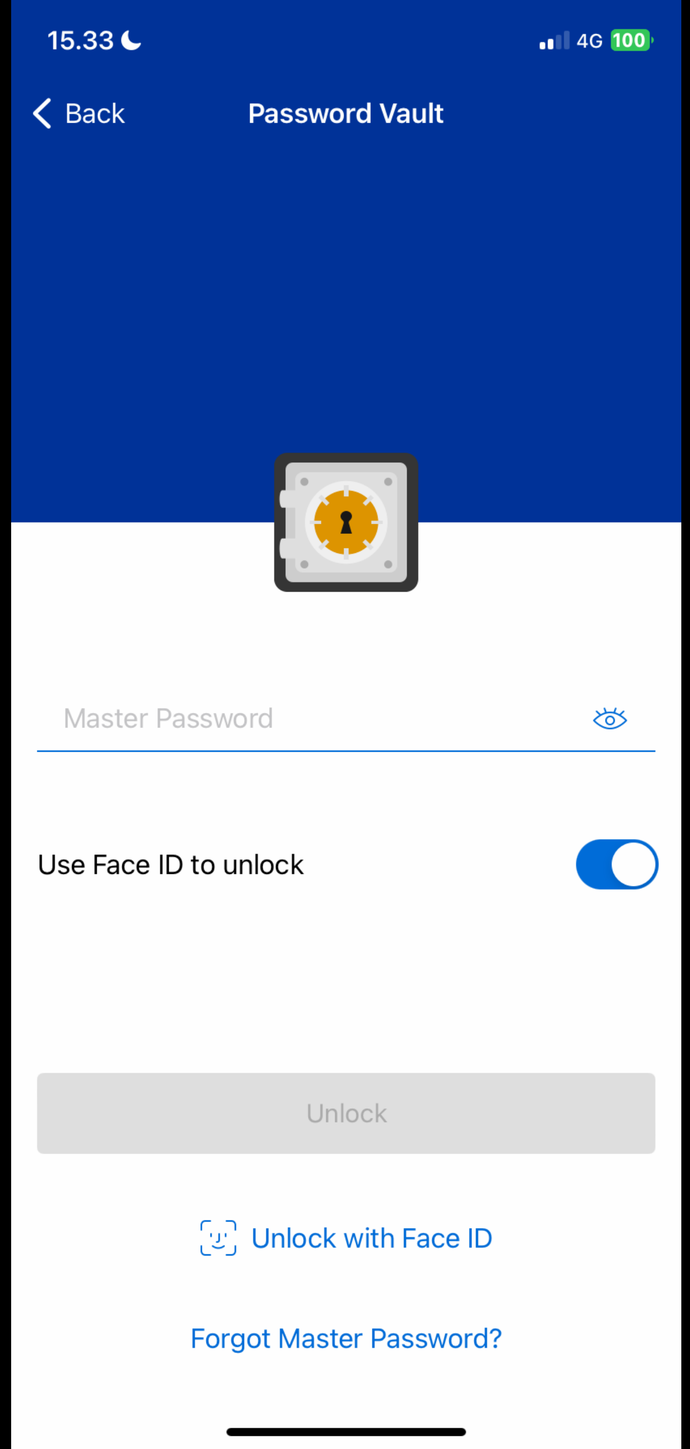
Adding a Keychain Group

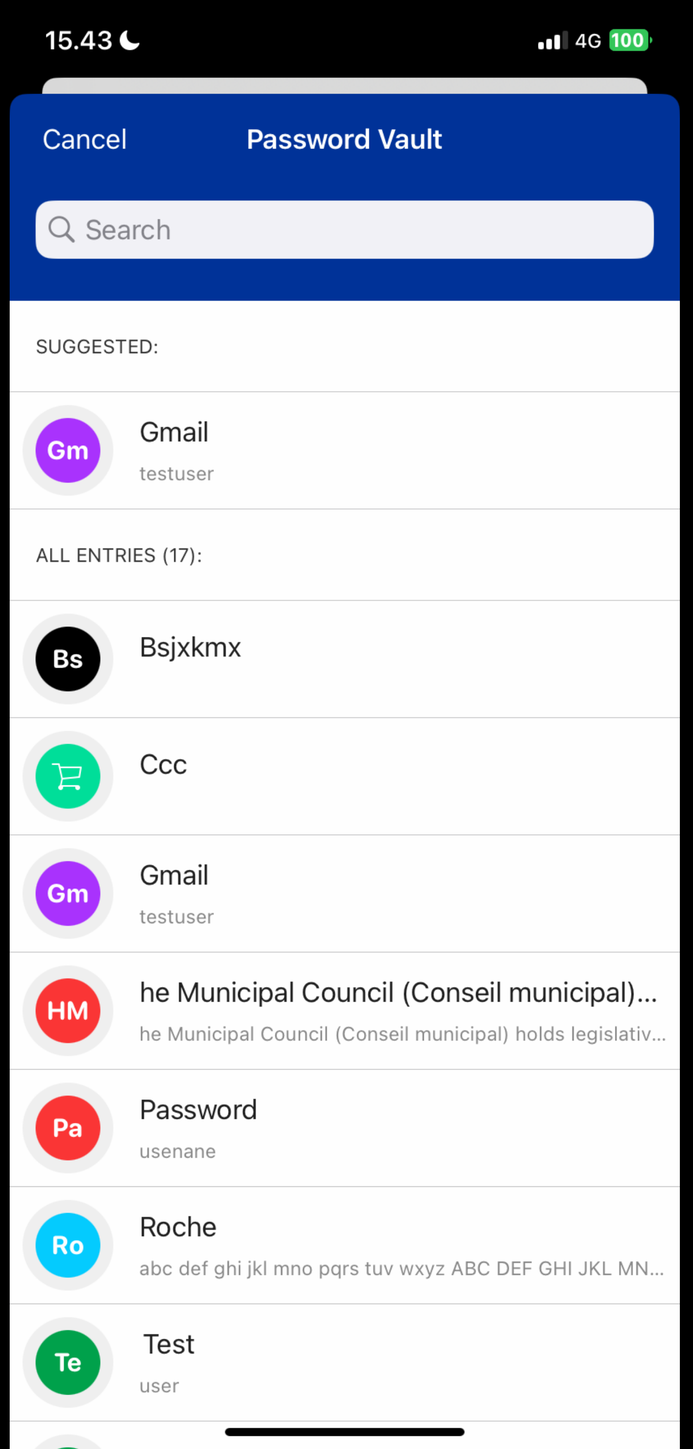


Keychain Sharing

Apps can share items now




15.33
4G
100
Back
Password Vault
Master Password
Use Face ID to unlock
Unlock
Unlock with Face ID
Forgot Master Password?


15.43
4G
100
Cancel
Password Vault
Search
SUGGESTED:
Gm
Gmail
testuser
ALL ENTRIES (17):
Bs
Bsjxkmx
Ccc
Gm
Gmail
testuser
HM
he Municipal Council (Conseil municipal)...
he Municipal Council (Conseil municipal) holds legislativ...
Pa
Password
usenane
Ro
Roche
abc def ghi jkl mno pqrs tuv wxyz ABC DEF GHI JKL MN...
Te
Test
user

Keychain Sharing

```
public final class Keychain {

    private let name: String?
    private let appGroup: String?

    public init(name: String?, appGroup: String? = nil) {
        self.name = name
        self.appGroup = appGroup
    }
}
```

Keychain Sharing

```
public final class Keychain {

    private let name: String?
    private let appGroup: String?

    public init(name: String?, appGroup: String? = nil) {
        self.name = name
        self.appGroup = appGroup
    }
}
```

Keychain Sharing

```
private var baseQuery: [String: Any] {
    var query = [String: Any]()
    query[kSecClass as String] = kSecClassGenericPassword
    if let name = name {
        query[kSecAttrService as String] = name
    }
    if let appGroup = appGroup {
        query[kSecAttrAccessGroup as String] = appGroup
    }
    return query
}
```

Keychain Sharing

```
private var baseQuery: [String: Any] {
    var query = [String: Any]()
    query[kSecClass as String] = kSecClassGenericPassword
    if let name = name {
        query[kSecAttrService as String] = name
    }
    if let appGroup = appGroup {
        query[kSecAttrAccessGroup as String] = appGroup
    }
    return query
}
```

Conclusion

- Keychain is a very powerful API…
- Although too low-level

If keychain wrappers don’t work…
Make your own :)

Q&A