Protecting the User's Data

The part you play

Session 714

Andrew R. Whalley
Imperial Security

Conrad Sauerwald
Metric Security

Michael Brouwer
Metric Security

Introduction

Introduction

- Platform Security team

Introduction

- Platform Security team
  - Any time a key is used to protect user data

Introduction

- Platform Security team
  - Any time a key is used to protect user data
    - Data Protection, Keychain, Secure Transport, CMS

Introduction

- Platform Security team
  - Any time a key is used to protect user data
    - Data Protection, Keychain, Secure Transport, CMS
  - Design and build solutions for internal clients

Introduction

- Platform Security team
  - Any time a key is used to protect user data
    - Data Protection, Keychain, Secure Transport, CMS
  - Design and build solutions for internal clients
  - High-level API for third party applications

Introduction

- Platform Security team
  - Any time a key is used to protect user data
    - Data Protection, Keychain, Secure Transport, CMS
  - Design and build solutions for internal clients
  - High-level API for third party applications
    - Security.framework, CommonCrypto

Introduction

- Platform Security team
  - Any time a key is used to protect user data
    - Data Protection, Keychain, Secure Transport, CMS
  - Design and build solutions for internal clients
  - High-level API for third party applications
    - Security.framework, CommonCrypto
  - Functionality exposed by even higher-level APIs

Introduction

- Platform Security team
  - Any time a key is used to protect user data
    - Data Protection, Keychain, Secure Transport, CMS
  - Design and build solutions for internal clients
  - High-level API for third party applications
    - Security.framework, CommonCrypto
  - Functionality exposed by even higher-level APIs
    - NSFileManager, CFNetwork

What We Will Demonstrate

What We Will Demonstrate

- A common situation

What We Will Demonstrate

- A common situation
  - Client app

What We Will Demonstrate

- A common situation
  - Client app
  - Web service

What We Will Demonstrate

- A common situation
  - Client app
  - Web service
- Hostile environment

What We Will Demonstrate

- A common situation
  - Client app
  - Web service
- Hostile environment
- Show

What We Will Demonstrate

- A common situation
  - Client app
  - Web service
- Hostile environment
- Show
  - What can happen

What We Will Demonstrate

- A common situation
  - Client app
  - Web service
- Hostile environment
- Show
  - What can happen
  - Why it matters

What We Will Demonstrate

- A common situation
  - Client app
  - Web service
- Hostile environment
- Show
  - What can happen
  - Why it matters
  - How you can avoid it

What We're Not Going to Cover

What We're Not Going to Cover

- Securing your app from exploitation

What We're Not Going to Cover

- Securing your app from exploitation
  - Sandboxing and hardening

What We're Not Going to Cover

- Securing your app from exploitation
  - Sandboxing and hardening
  - Secure coding practices

What We're Not Going to Cover

- Securing your app from exploitation
  - Sandboxing and hardening
  - Secure coding practices
- Cryptographic protocol design

What We're Not Going to Cover

- Securing your app from exploitation
  - Sandboxing and hardening
  - Secure coding practices
- Cryptographic protocol design
- Digital Rights Management

What We're Not Going to Cover

- Securing your app from exploitation
  - Sandboxing and hardening
  - Secure coding practices
- Cryptographic protocol design
- Digital Rights Management
  - Not protecting the user's data

Everything Is User Data

Everything Is User Data

- Documents

Everything Is User Data

- Documents
  - Text

Everything Is User Data

- Documents
  - Text
  - Photos

Everything Is User Data

- Documents
  - Text
  - Photos
- Credentials

Everything Is User Data

- Documents
  - Text
  - Photos
- Credentials
- Preferences

Everything Is User Data

- Documents
  - Text
  - Photos
- Credentials
- Preferences
- Other

Everything Is User Data

- Documents
  - Text
  - Photos
- Credentials
- Preferences
- Other
  - Photo metadata

Everything Is User Data

- Documents
  - Text
  - Photos
- Credentials
- Preferences
- Other
  - Photo metadata
  - Unique identifiers

User Data Is Everywhere

User Data Is Everywhere

- On the device

User Data Is Everywhere

- On the device
- In transit

User Data Is Everywhere

- On the device
- In transit
  - Network data

User Data Is Everywhere

- On the device
- In transit
  - Network data
- Off the device

User Data Is Everywhere

- On the device
- In transit
  - Network data
- Off the device
  - Your server

User Data Is Everywhere

- On the device
- In transit
  - Network data
- Off the device
  - Your server
  - Backup

User Data Is Everywhere

- On the device
- In transit
  - Network data
- Off the device
  - Your server
  - Backup
  - iCloud

Naivete.app

Simple media client

Conrad Sauerwald

Naivete.app

Simple media client

- Downloads images from a server
- Can view and cache locally
- Source code available

Naivete.app Networking

Application uses HTTP

```
NSString *server_url = @”http://server.com/streams/arw”;
NSURLRequest *request = [NSURLRequest requestWithURL:
                         [NSURL URLWithString: server_url ]];
```

Naivete.app Networking

Application uses HTTP

```
NSString *server_url = @"http://server.com/streams/arw";
NSURLRequest *request = [NSURLRequest requestWithURL:
                         [NSURL URLWithString: server_url ]];

[[NSURLConnection alloc] initWithRequest: request delegate: self];
```

Naivete.app Networking

Application uses HTTP

```
NSString *server_url = @"http://server.com/streams/arw";
NSURLRequest *request = [NSURLRequest requestWithURL:
                         [NSURL URLWithString: server_url ]];

[[NSURLConnection alloc] initWithRequest: request delegate: self];

@interface MainViewController : UIViewController <NSURLConnectionDelegate>
```

Naivete.app Networking

Application uses HTTP

```
NSString *server_url = @"http://server.com/streams/arw";
NSURLRequest *request = [NSURLRequest requestWithURL:
                         [NSURL URLWithString: server_url ]];

[[NSURLConnection alloc] initWithRequest: request delegate: self];

@interface MainViewController : UIViewController <NSURLConnectionDelegate>
 - (void)connection:(NSURLConnection *)c didReceiveData:(NSData *)data
```

Naivete.app Networking

Application uses HTTP

```
NSString *server_url = @"http://server.com/streams/arw";
NSURLRequest *request = [NSURLRequest requestWithURL:
                         [NSURL URLWithString: server_url ]];

[[NSURLConnection alloc] initWithRequest: request delegate: self];

@interface MainViewController : UIViewController <NSURLConnectionDelegate>
 - (void)connection:(NSURLConnection *)c didReceiveData:(NSData *)data
 - (void)connection:(NSURLConnection *)c didFailWithError:(NSError *)error
```

Naivete.app Networking

Application uses HTTP

```
NSString *server_url = @"http://server.com/streams/arw";
NSURLRequest *request = [NSURLRequest requestWithURL:
                         [NSURL URLWithString: server_url ]];

[[NSURLConnection alloc] initWithRequest: request delegate: self];

@interface MainViewController : UIViewController <NSURLConnectionDelegate>
 - (void)connection:(NSURLConnection *)c didReceiveData:(NSData *)data
 - (void)connection:(NSURLConnection *)c didFailWithError:(NSError *)error
 - (void)connectionDidFinishLoading:(NSURLConnection *)c
@end
```

Naivete.app Authentication

User authenticates to server

```
- (void)connection:(NSURLConnection *)c
     willSendRequestForAuthenticationChallenge:
          (NSURLAuthenticationChallenge *)ch
```

Naivete.app Authentication

User authenticates to server

```
- (void)connection:(NSURLConnection *)c
     willSendRequestForAuthenticationChallenge:
          (NSURLAuthenticationChallenge *)ch

NSString * const NSURLAuthenticationMethodDefault;
```

Naivete.app Authentication

User authenticates to server

```
- (void)connection:(NSURLConnection *)c
     willSendRequestForAuthenticationChallenge:
          (NSURLAuthenticationChallenge *)ch

NSString * const NSURLAuthenticationMethodDefault;

NSURLCredential *cred = [NSURLCredential credentialWithUser:[s username]
                                 password:[s password]
                                 persistence:NSURLCredentialPersistenceNone];
[[ch sender] useCredential:cred forAuthenticationChallenge:ch];
```

Naivete.app Authentication

User authenticates to server

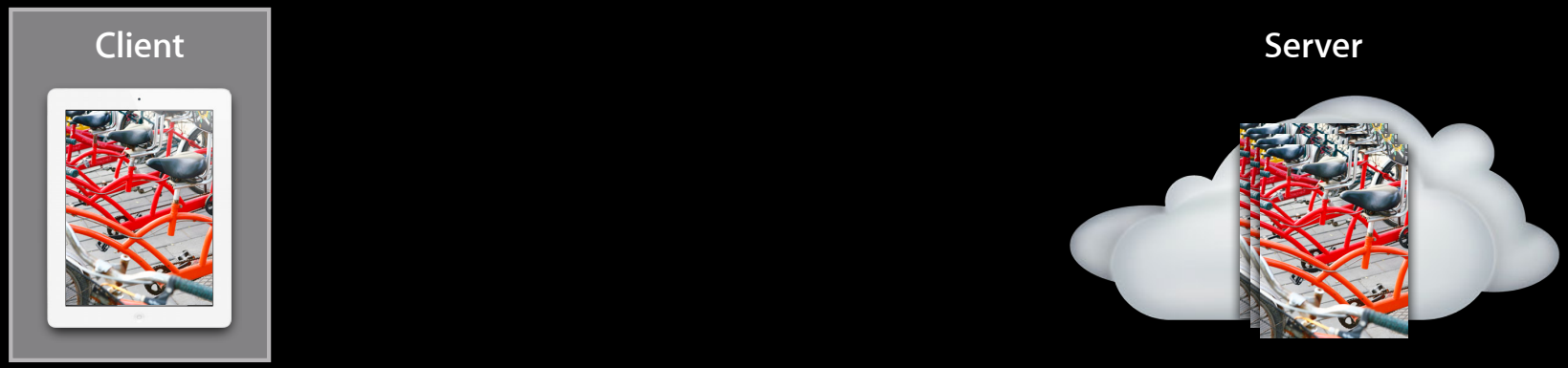
```
- (void)connection:(NSURLConnection *)c
     willSendRequestForAuthenticationChallenge:
          (NSURLAuthenticationChallenge *)ch

NSString * const NSURLAuthenticationMethodDefault;

NSURLCredential *cred = [NSURLCredential credentialWithUser:[s username]
                                 password:[s password]
                                 persistence:NSURLCredentialPersistenceNone];
[[ch sender] useCredential:cred forAuthenticationChallenge:ch];

enum { NSURLCredentialPersistencePermanent, … };
```

Demo

Andrew Whalley

Network Interception

What does the attacker see?

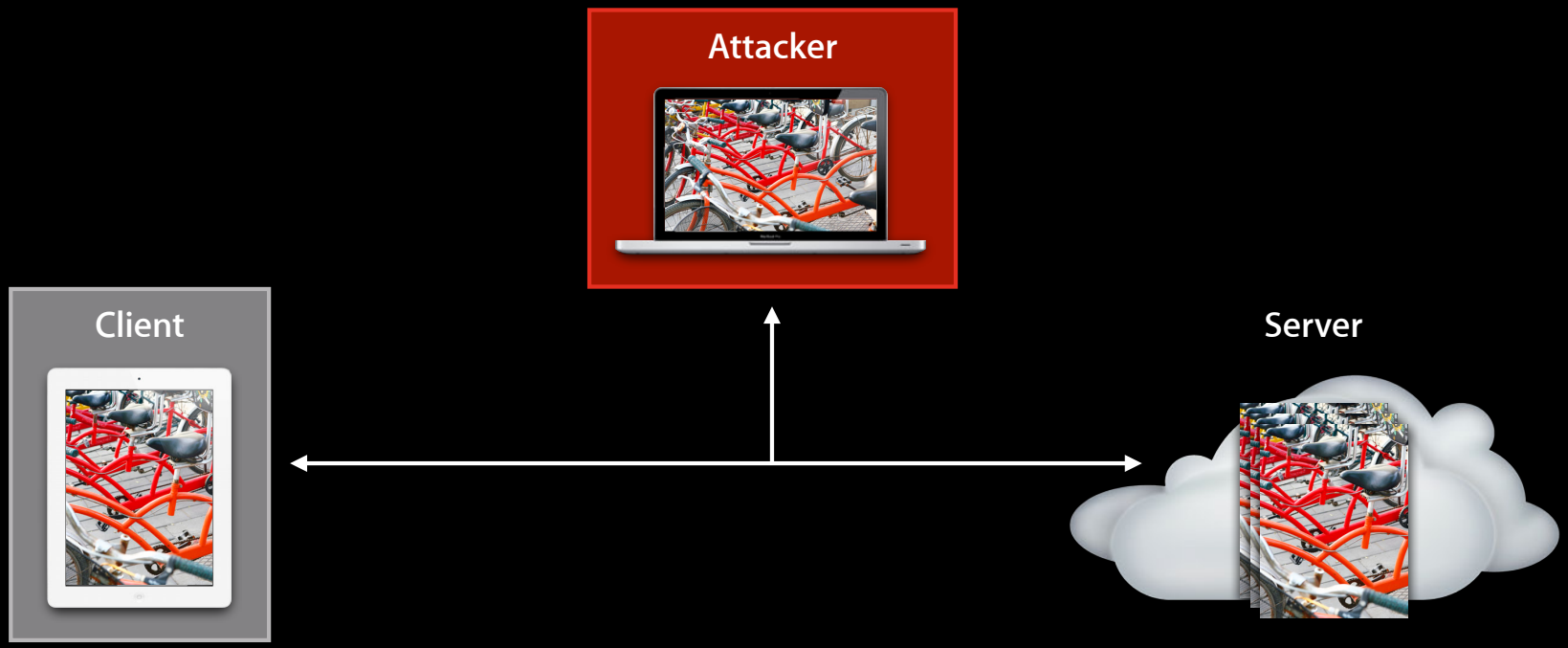


Network Interception

What does the attacker see?

Network Interception

What does the attacker see?



Why Secure Naivete.app?

Why Secure Naivete.app?

- Users are entrusting their data to your app

Why Secure Naivete.app?

- Users are entrusting their data to your app
  - It could be sensitive

Why Secure Naivete.app?

- Users are entrusting their data to your app
  - It could be sensitive
  - Assume it is and protect it

Why Secure Naivete.app?

- Users are entrusting their data to your app
  - It could be sensitive
  - Assume it is and protect it
- Real threats

Why Secure Naivete.app?

- Users are entrusting their data to your app
  - It could be sensitive
  - Assume it is and protect it
- Real threats
  - Public networks

Why Secure Naivete.app?

- Users are entrusting their data to your app
  - It could be sensitive
  - Assume it is and protect it
- Real threats
  - Public networks
  - Lost device

Why Secure Naivete.app?

- Users are entrusting their data to your app
  - It could be sensitive
  - Assume it is and protect it
- Real threats
  - Public networks
  - Lost device
- Credentials are valuable

Securing Naivete.app

Protecting data in transit

Conrad Sauerwald

Weak Password Authentication

How to improve it

Weak Password Authentication

How to improve it

- Don’t send the password in the clear
  - Challenge/response sends password derived authenticator
  - Change `NSURLAuthenticationMethodHTTPBasic` to `NSURLAuthenticationMethodHTTPDigest`

Weak Password Authentication

How to improve it

- Don’t send the password in the clear
  - Challenge/response sends password derived authenticator
  - Change `NSURLAuthenticationMethodHTTPBasic` to `NSURLAuthenticationMethodHTTPDigest`
- Cleartext password needed during signup

Weak Password Authentication

How to improve it

- Don’t send the password in the clear
  - Challenge/response sends password derived authenticator
  - Change `NSURLAuthenticationMethodHTTPBasic` to `NSURLAuthenticationMethodHTTPDigest`
- Cleartext password needed during signup
- Weak password can be brute forced

Weak Password Authentication

How to improve it

- Don’t send the password in the clear
  - Challenge/response sends password derived authenticator
  - Change `NSURLAuthenticationMethodHTTPBasic` to `NSURLAuthenticationMethodHTTPDigest`
- Cleartext password needed during signup
- Weak password can be brute forced
- More advanced alternatives
  - Certificates, SRP

Data on the Wire

Encryption and authentication

Data on the Wire

Encryption and authentication

- Transport Layer Security (TLS)
  - Provides confidentiality and integrity
  - Evolution

Data on the Wire

Encryption and authentication

- Transport Layer Security (TLS)
  - Provides confidentiality and integrity
  - Evolution
- Server authentication
  - Talking to the right server

Data on the Wire

Encryption and authentication

- Transport Layer Security (TLS)
  - Provides confidentiality and integrity
  - Evolution
- Server authentication
  - Talking to the right server
- Data encryption
  - Cannot be deciphered by attacker

Data on the Wire

Encryption and authentication

- Transport Layer Security (TLS)
  - Provides confidentiality and integrity
  - Evolution
- Server authentication
  - Talking to the right server
- Data encryption
  - Cannot be deciphered by attacker
- Data integrity
  - Modification by attacker is detected

TLS Server Authentication with Certificates

Honest Abe's Cheap Bicycles and Lightly Used Certificates

TLS Server Authentication with Certificates

Honest Abe's Cheap Bicycles and Lightly Used Certificates

TLS Server Authentication with Certificates

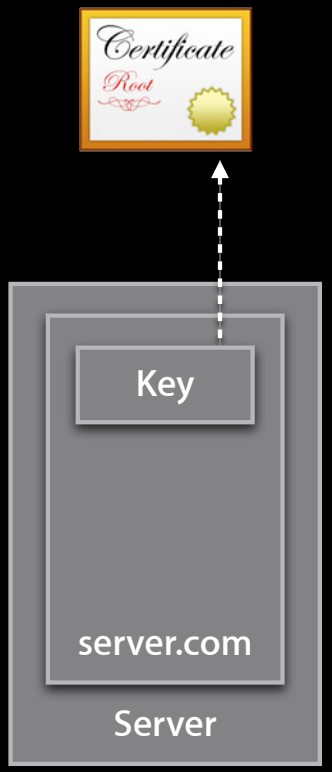
Honest Abe's Cheap Bicycles and Lightly Used Certificates

TLS Server Authentication with Certificates

Honest Abe's Cheap Bicycles and Lightly Used Certificates

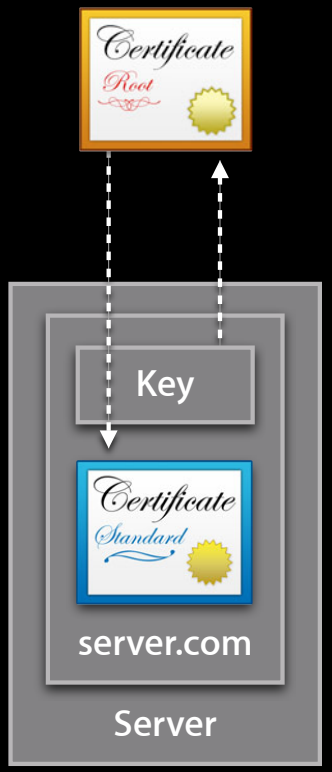
TLS Server Authentication with Certificates

Honest Abe's Cheap Bicycles and Lightly Used Certificates



TLS Server Authentication with Certificates

Honest Abe's Cheap Bicycles and Lightly Used Certificates

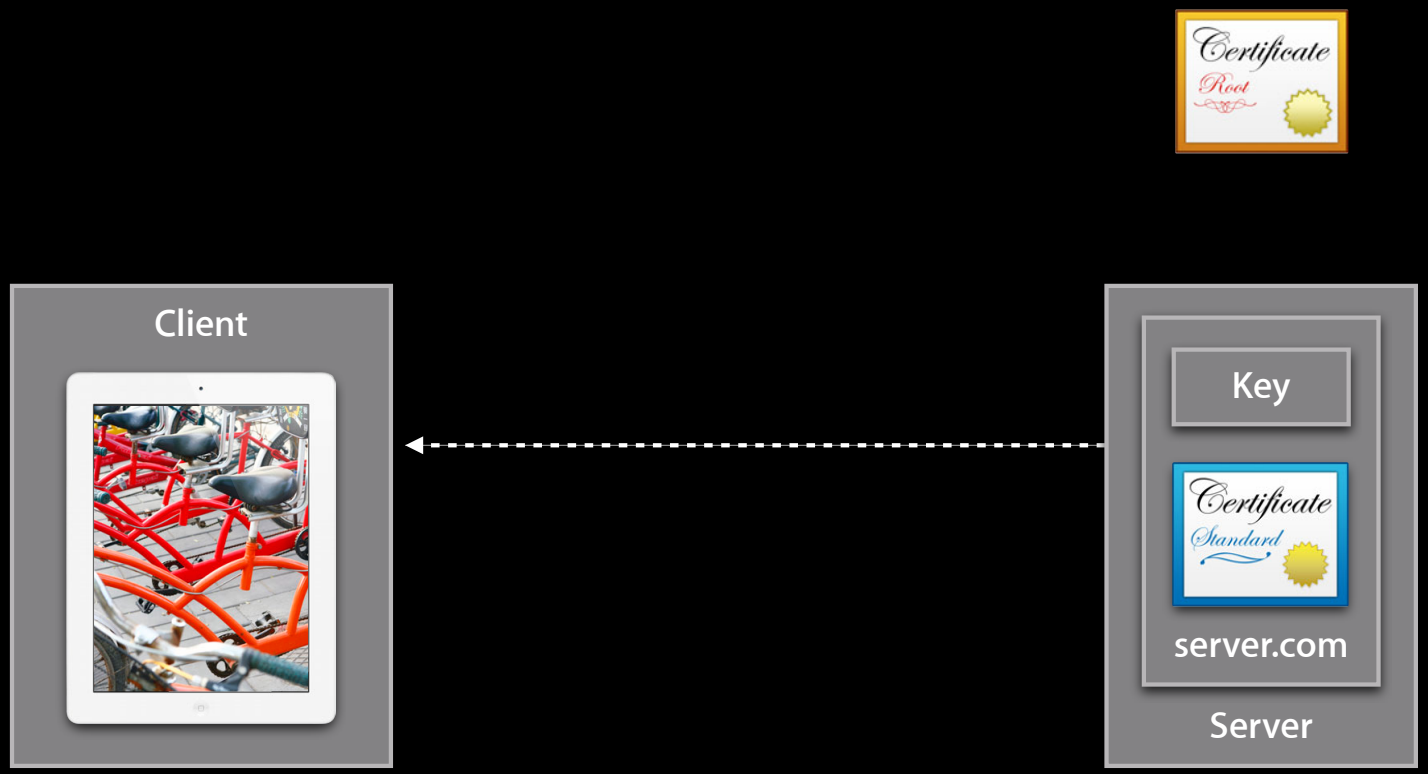


TLS Server Authentication with Certificates

Honest Abe's Cheap Bicycles and Lightly Used Certificates

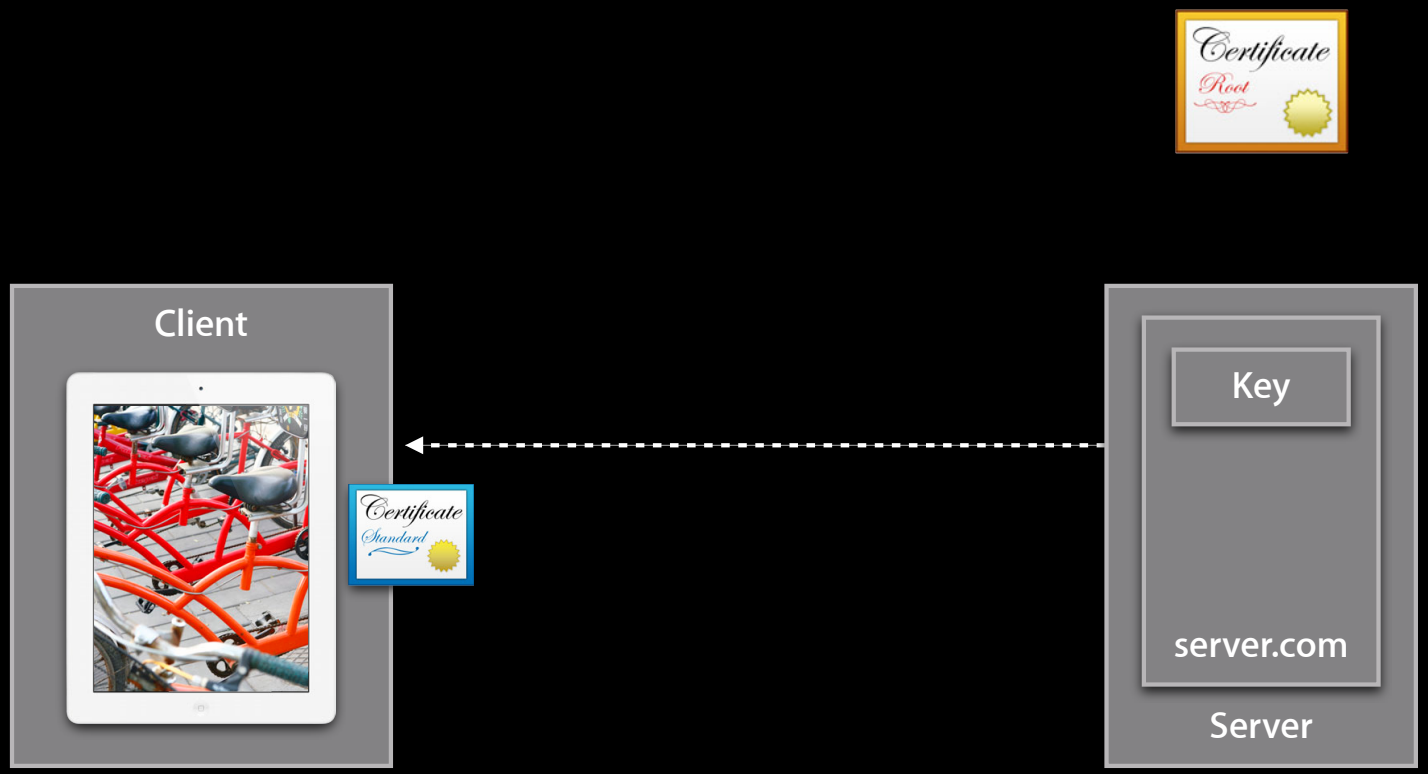
TLS Server Authentication with Certificates

Honest Abe's Cheap Bicycles and Lightly Used Certificates



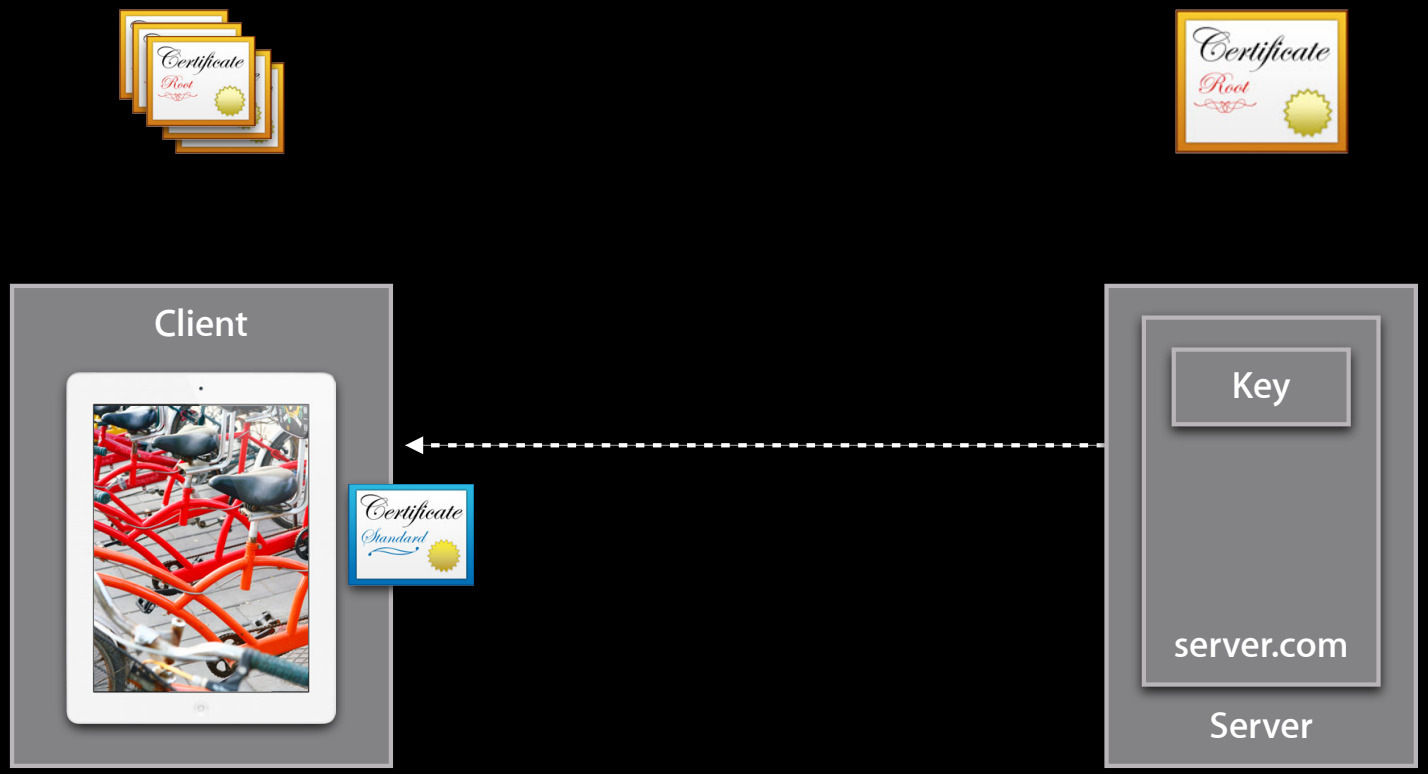
TLS Server Authentication with Certificates

Honest Abe's Cheap Bicycles and Lightly Used Certificates



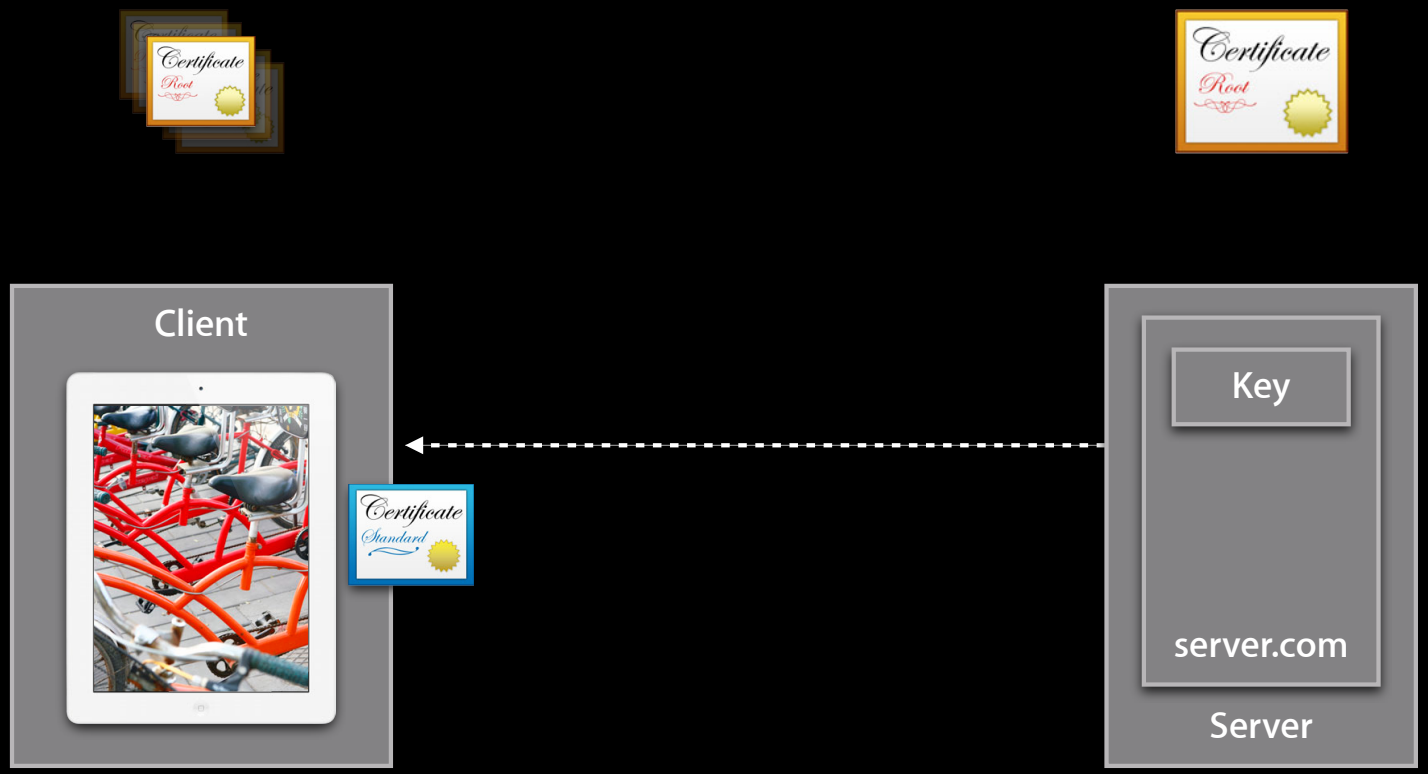
TLS Server Authentication with Certificates

Honest Abe's Cheap Bicycles and Lightly Used Certificates



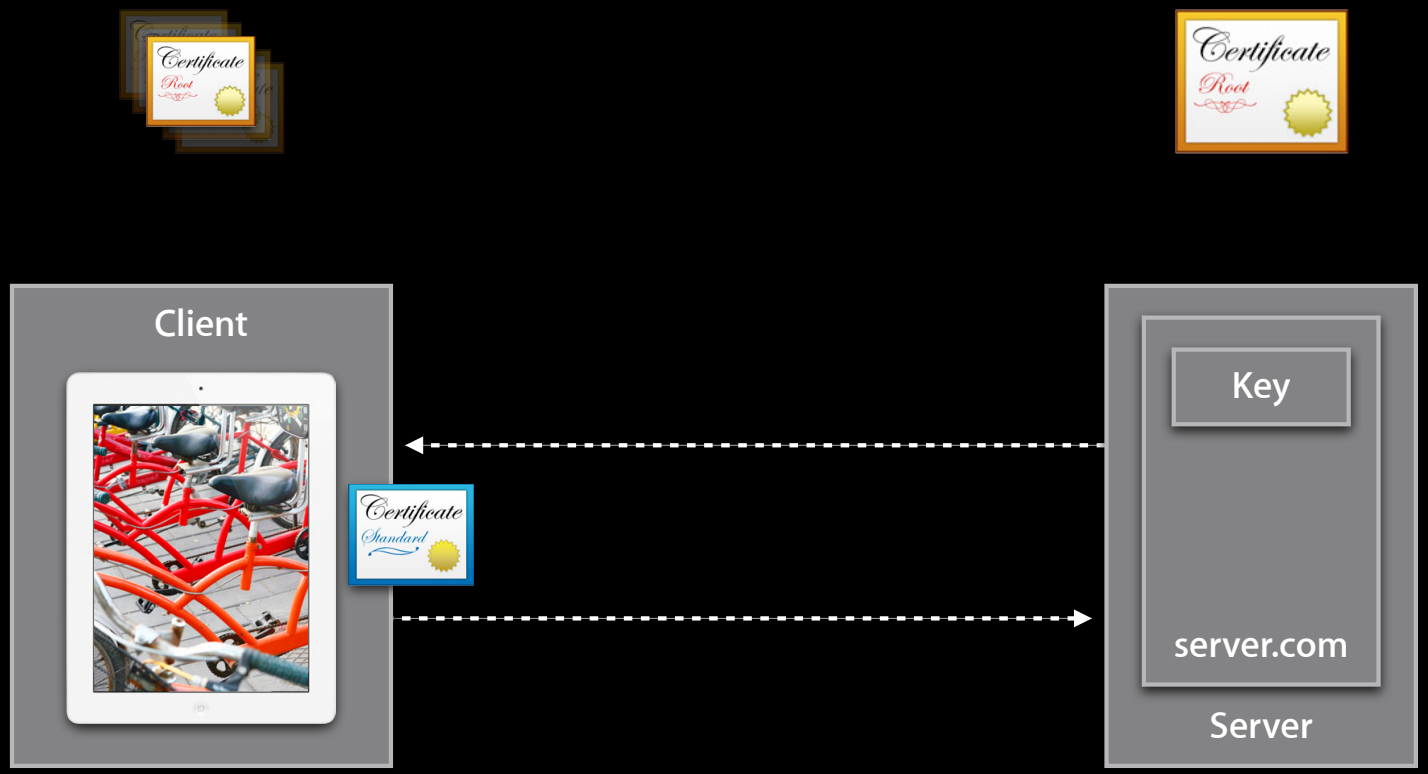
TLS Server Authentication with Certificates

Honest Abe's Cheap Bicycles and Lightly Used Certificates



TLS Server Authentication with Certificates

Honest Abe's Cheap Bicycles and Lightly Used Certificates



Certificates

In a nutshell

Certificates

In a nutshell

- Certificate
  - Identity (i.e. hostname)
  - Verification key

Certificates

In a nutshell

- Certificate
  - Identity (i.e. hostname)
  - Verification key
- Authority
  - Issuer
  - Signature

Certificates

In a nutshell

- Certificate
  - Identity (i.e. hostname)
  - Verification key
- Authority
  - Issuer
  - Signature
- Tree of certificates
  - Authorities at the bottom
  - Leafs at the top

Certificates

In a nutshell

- Certificate
  - Identity (i.e. hostname)
  - Verification key
- Authority
  - Issuer
  - Signature
- Tree of certificates
  - Authorities at the bottom
  - Leafs at the top
- Prearranged trust of authorities

Using Transport Layer Security (TLS)

Foundation client

Using Transport Layer Security (TLS)

Foundation client

- One letter change

```
// “After” code
NSString *server_url = @“https://server.com/streams/arw/”;

/*
   connect exactly as before
*/
```

What's Next?

Because nothing is that simple

What's Next?

Because nothing is that simple

- Maybe it is… Profit!

What's Next?

Because nothing is that simple

- Maybe it is… Profit!
- Customize trust
  - Self-signed certificates for testing
  - Limiting to private authorities

What's Next?

Because nothing is that simple

- Maybe it is… Profit!
- Customize trust
  - Self-signed certificates for testing
  - Limiting to private authorities
- Client authentication with certificates

What's Next?

Because nothing is that simple

- Maybe it is… Profit!
- Customize trust
  - Self-signed certificates for testing
  - Limiting to private authorities
- Client authentication with certificates
- DTLS

What's Next?

Because nothing is that simple

- Maybe it is… Profit!
- Customize trust
  - Self-signed certificates for testing
  - Limiting to private authorities
- Client authentication with certificates
- DTLS
- More control?
  - Authentication challenges and SecTrust
  - SecureTransport

Demo

Andrew Whalley

Securing Naivete.app

Protecting data at rest

Michael Brouwer

Protecting Data on the Device

Last line of defense

Protecting Data on the Device

Last line of defense

- Jailbreaks and forensic tools allow filesystem access

Protecting Data on the Device

Last line of defense

- Jailbreaks and forensic tools allow filesystem access
- Provides protection in the case a device is lost or stolen

Protecting Data on the Device

Last line of defense

- Jailbreaks and forensic tools allow filesystem access
- Provides protection in the case a device is lost or stolen
- Data Protection allows data to be tied to the user's passcode

Data Protection

Versus full disk encryption

Data Protection

Versus full disk encryption

- Data Protection also ties data to a device

Data Protection

Versus full disk encryption

- Data Protection also ties data to a device
- Volume based vs. File based

Data Protection

Versus full disk encryption

- Data Protection also ties data to a device
- Volume based vs. File based
- iOS devices are almost always on

Data Protection

Versus full disk encryption

- Data Protection also ties data to a device
- Volume based vs. File based
- iOS devices are almost always on
- Different classes of data available at different times

Data Protection

Versus full disk encryption

- Data Protection also ties data to a device
- Volume based vs. File based
- iOS devices are almost always on
- Different classes of data available at different times
  - After first unlock (FDE like)

Data Protection

Versus full disk encryption

- Data Protection also ties data to a device
- Volume based vs. File based
- iOS devices are almost always on
- Different classes of data available at different times
  - After first unlock (FDE like)
    - Keys remain in memory until power off

Data Protection

Versus full disk encryption

- Data Protection also ties data to a device
- Volume based vs. File based
- iOS devices are almost always on
- Different classes of data available at different times
  - After first unlock (FDE like)
    - Keys remain in memory until power off
  - While unlocked

Data Protection

Versus full disk encryption

- Data Protection also ties data to a device
- Volume based vs. File based
- iOS devices are almost always on
- Different classes of data available at different times
  - After first unlock (FDE like)
    - Keys remain in memory until power off
  - While unlocked
    - Keys purged 10 seconds after lock

Data Protection Key Hierarchy

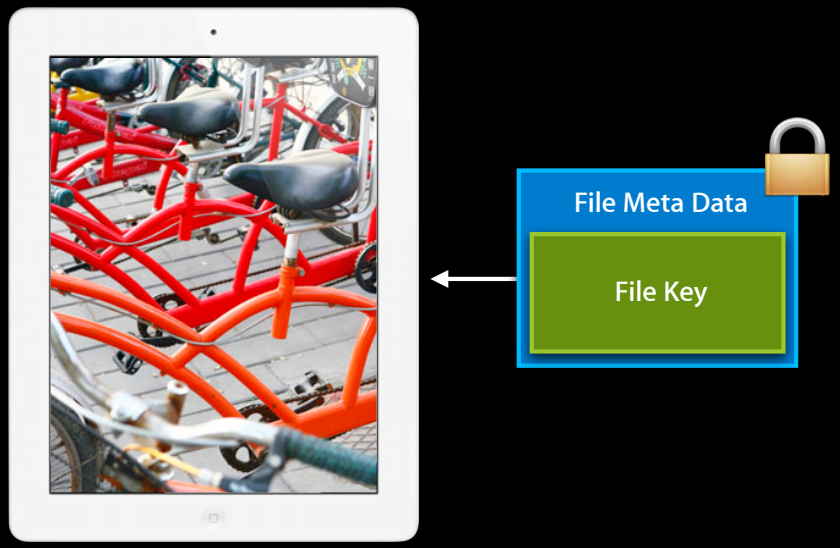
Protected file

Data Protection Key Hierarchy

Protected file

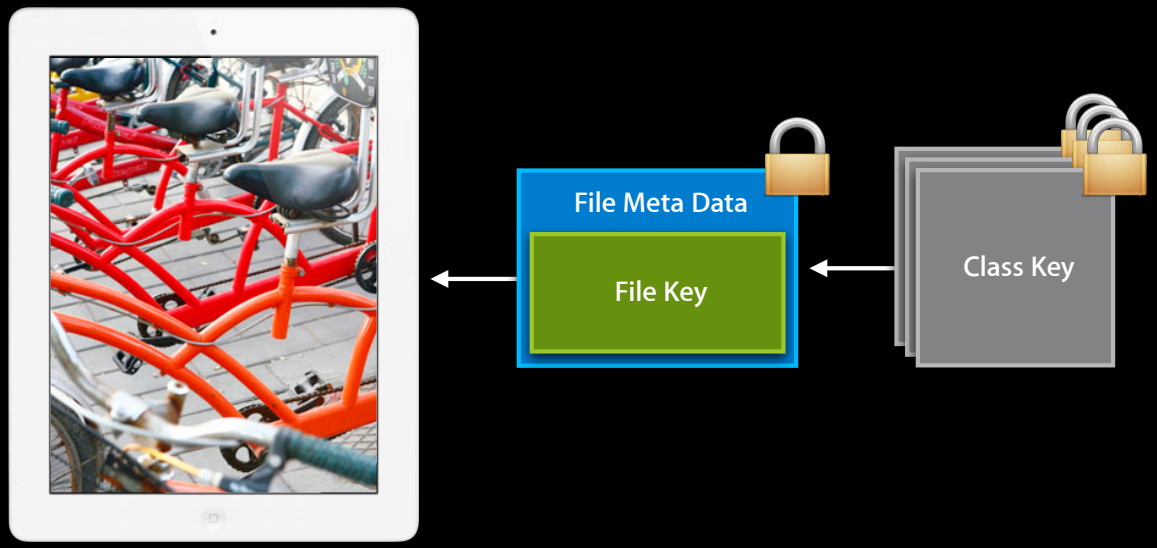
Data Protection Key Hierarchy

Protected file



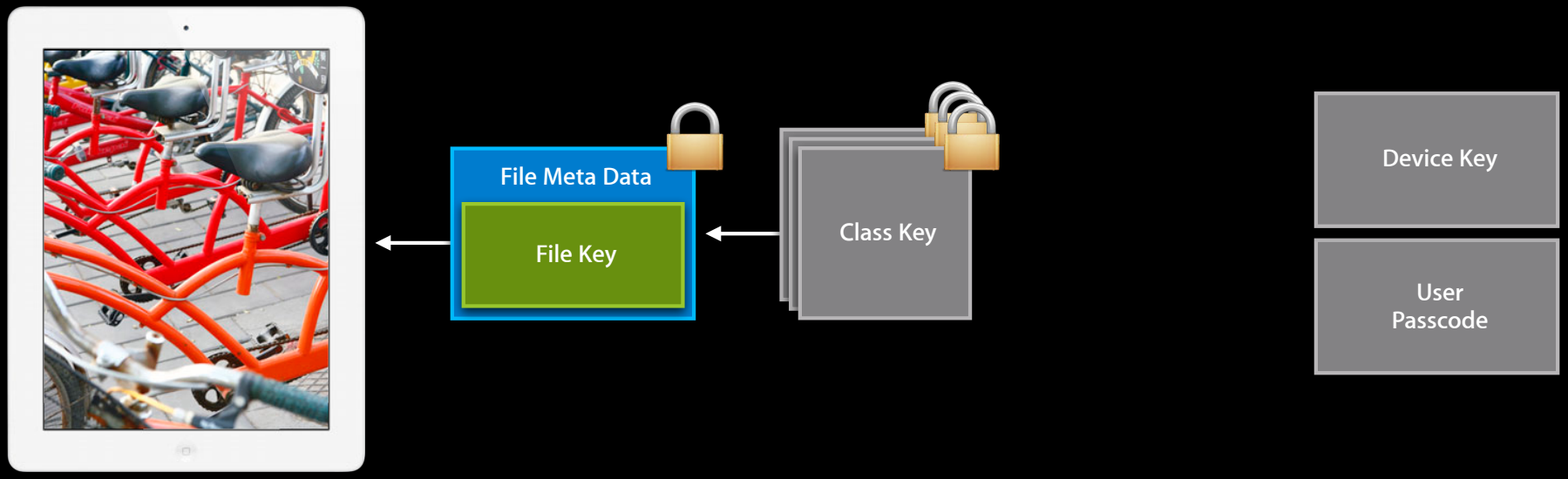
Data Protection Key Hierarchy

Protected file



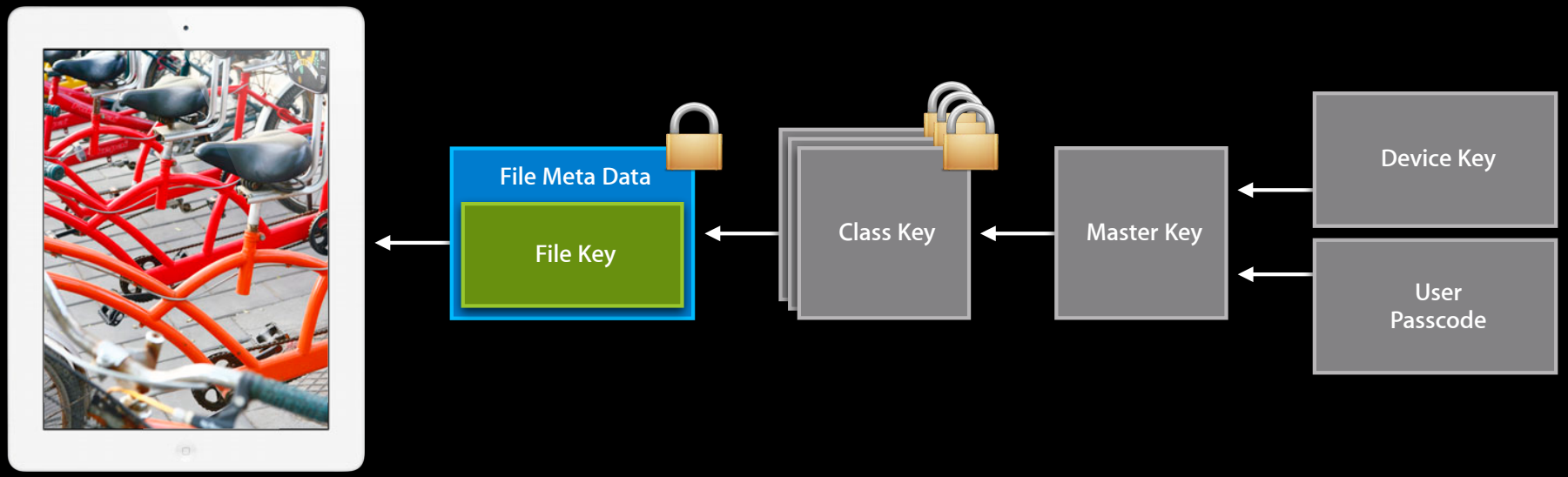
Data Protection Key Hierarchy

Protected file



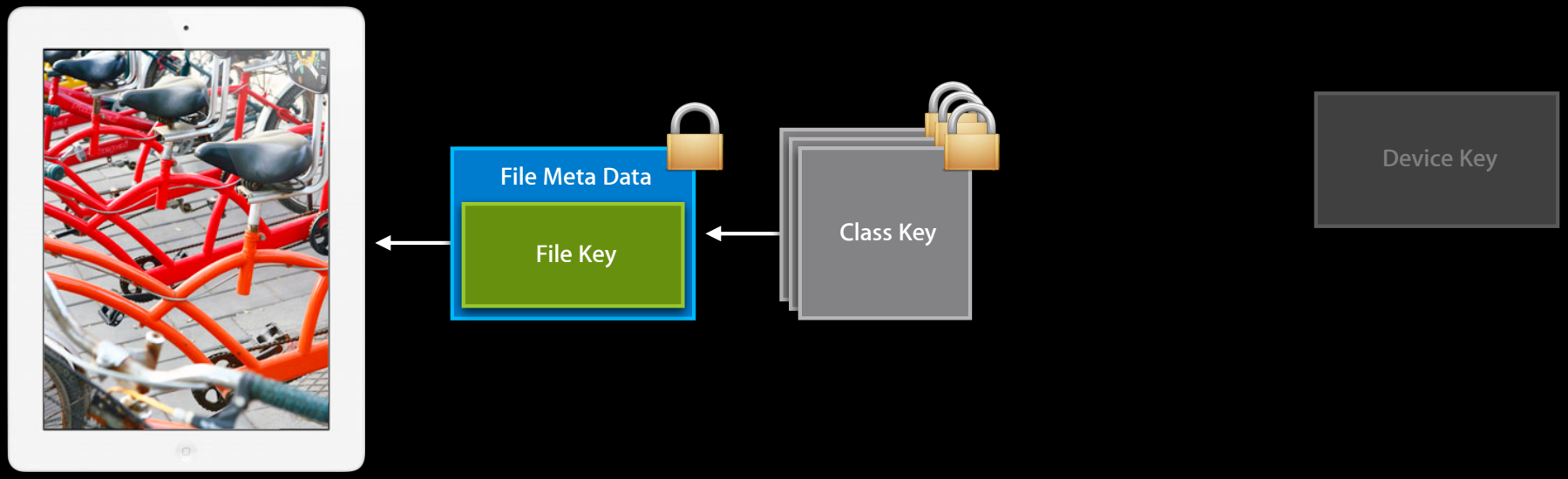
Data Protection Key Hierarchy

Protected file



Data Protection Key Hierarchy

Protected file



APIs Offering Data Protection

NSFileManager	`NSFileProtectionKey`	`NSFileProtection…`
CoreData	`NSFileProtectionKey`	`NSFileProtection…`
NSData	`NSDataWritingOptions`	`NSDataWritingFileProtection…`
sqlite3	`sqlite3_open_v2 option`	`SQLITE_OPEN_FILEPROTECTION_…`
SecItem	`kSecAttrAccessible`	`kSecAttrAccessible…`

APIs Offering Data Protection

NSFileManager	NSFileProtectionKey	NSFileProtection…
CoreData	NSFileProtectionKey	NSFileProtection…
NSData	NSDataWritingOptions	NSDataWritingFileProtection…
sqlite3	sqlite3_open_v2 option	SQLITE_OPEN_FILEPROTECTION_…
SecItem	kSecAttrAccessible	kSecAttrAccessible…

APIs Offering Data Protection

NSFileManager	NSFileProtectionKey	NSFileProtection…
CoreData	NSFileProtectionKey	NSFileProtection…
NSData	NSDataWritingOptions	NSDataWritingFileProtection…
sqlite3	sqlite3_open_v2 option	SQLITE_OPEN_FILEPROTECTION_…
SecItem	kSecAttrAccessible	kSecAttrAccessible…

APIs Offering Data Protection

NSFileManager	NSFileProtectionKey	NSFileProtection…
CoreData	NSFileProtectionKey	NSFileProtection…
NSData	NSDataWritingOptions	NSDataWritingFileProtection…
sqlite3	sqlite3_open_v2 option	SQLITE_OPEN_FILEPROTECTION_…
SecItem	kSecAttrAccessible	kSecAttrAccessible…

APIs Offering Data Protection

NSFileManager	`NSFileProtectionKey`	`NSFileProtection…`
CoreData	`NSFileProtectionKey`	`NSFileProtection…`
NSData	`NSDataWritingOptions`	`NSDataWritingFileProtection…`
sqlite3	`sqlite3_open_v2 option`	`SQLITE_OPEN_FILEPROTECTION_…`
SecItem	`kSecAttrAccessible`	`kSecAttrAccessible…`

APIs Offering Data Protection

NSFileManager	`NSFileProtectionKey`	`NSFileProtection…`
CoreData	`NSFileProtectionKey`	`NSFileProtection…`
NSData	`NSDataWritingOptions`	`NSDataWritingFileProtection…`
sqlite3	`sqlite3_open_v2 option`	`SQLITE_OPEN_FILEPROTECTION_…`
SecItem	`kSecAttrAccessible`	`kSecAttrAccessible…`

Data Only Available When Unlocked

FileProtectionComplete

```
-(BOOL)writeImage:(UIImage *)image toPath:path error:(NSError **)error
{
        NSData *data = UIImageJPEGRepresentation(image, 1.0);
        return [data writeToFile:path options:
            NSDataWritingFileProtectionComplete error:error];
}
```

Data Only Available When Unlocked

Considerations

- Only as good as the passcode
- Cannot access when locked
  - Use `NSFileProtectionCompleteUnlessOpen`
  - Upgrade to `NSFileProtectionComplete` when unlocked

Data Dropbox

FileProtectionCompleteUnlessOpen

```
-(BOOL)writeImageWhileLocked:(UIImage *)image toPath:path
                       error:(NSError **)error
{
        NSData *data = UIImageJPEGRepresentation(image, 1.0);
        return [data writeToFile: path
                         options: NSDataWritingFileProtectionComplete
                           error: error]
            || [data writeToFile: path
                         options: NSDataWritingFileProtectionCompleteUnlessOpen
                           error: error];
}
```

Upgrade to FileProtectionComplete

```
-(void)upgradeImagesInDir:(NSString *)dir error:(NSError **)error {
    NSFileManager *fm = [NSFileManager defaultManager];
    NSDirectoryEnumerator *de = [fm enumeratorAtPath: dir];
    for (NSString *path in de) {
         NSDictionary *attrs = [de fileAttributes];
         if (![[attrs objectForKey: NSFileProtectionKey]
            isEqual: NSFileProtectionComplete]) {
            attrs = [NSDictionary dictionaryWithObject:
                NSFileProtectionComplete forKey: NSFileProtectionKey];
            [fm setAttributes: attrs ofItemAtPath: path error: error];
        }
    }
}
```

Upgrade to FileProtectionComplete

```
-(void)upgradeImagesInDir:(NSString *)dir error:(NSError **)error {
    NSFileManager *fm = [NSFileManager defaultManager];
    NSDirectoryEnumerator *de = [fm enumeratorAtPath: dir];
    for (NSString *path in de) {
        NSDictionary *attrs = [de fileAttributes];
        if (![[attrs objectForKey: NSFileProtectionKey]
           isEqual: NSFileProtectionComplete]) {
           attrs = [NSDictionary dictionaryWithObject:
               NSFileProtectionComplete forKey: NSFileProtectionKey];
           [fm setAttributes: attrs ofItemAtPath: path error: error];
        }
    }
}
```

Upgrade to FileProtectionComplete

```
-(void)upgradeImagesInDir:(NSString *)dir error:(NSError **)error {
    NSFileManager *fm = [NSFileManager defaultManager];
    NSDirectoryEnumerator *de = [fm enumeratorAtPath: dir];
    for (NSString *path in de) {
        NSDictionary *attrs = [de fileAttributes];
        if (![[attrs objectForKey: NSFileProtectionKey]
            isEqual: NSFileProtectionComplete]) {
            attrs = [NSDictionary dictionaryWithObject:
                NSFileProtectionComplete forKey: NSFileProtectionKey];
            [fm setAttributes: attrs ofItemAtPath: path error: error];
        }
    }
}
```

Upgrade to FileProtectionComplete

```
-(void)upgradeImagesInDir:(NSString *)dir error:(NSError **)error {
    NSFileManager *fm = [NSFileManager defaultManager];
    NSDirectoryEnumerator *de = [fm enumeratorAtPath: dir];
    for (NSString *path in de) {
         NSDictionary *attrs = [de fileAttributes];
         if (![[attrs objectForKey: NSFileProtectionKey]
            isEqual: NSFileProtectionComplete]) {
            attrs = [NSDictionary dictionaryWithObject:
                NSFileProtectionComplete forKey: NSFileProtectionKey];
            [fm setAttributes: attrs ofItemAtPath: path error: error];
        }
    }
}
```

Background Read

…ProtectionCompleteUntilFirstUserAuthentication

Background Read

…ProtectionCompleteUntilFirstUserAuthentication

- Also solves the problem of data access when locked

Background Read

…ProtectionCompleteUntilFirstUserAuthentication

- Also solves the problem of data access when locked
- Protects data from reboot until first unlock

Background Read

…ProtectionCompleteUntilFirstUserAuthentication

- Also solves the problem of data access when locked
- Protects data from reboot until first unlock
  - Better than default of none against attacks that require a reboot

Background Readable Database

…ProtectionCompleteUntilFirstUserAuthentication

```
-(int)openDatabase(const char *filename, sqlite3 **pdb)
{
     return sqlite3_open_v2(filename, pdb,
        SQLITE_OPEN_FILEPROTECTION_COMPLETEUNTILFIRSTUSERAUTHENTICATION,
        NULL);
}
```

Data Protection for All Files

Data Protection for All Files

- Add `com.apple.developer.default-data-protection` entitlement

Data Protection for All Files

- Add `com.apple.developer.default-data-protection` entitlement
- Use `NSFileProtectionComplete` as its value

Data Protection for All Files

- Add `com.apple.developer.default-data-protection` entitlement
- Use `NSFileProtectionComplete` as its value
- Sets the default

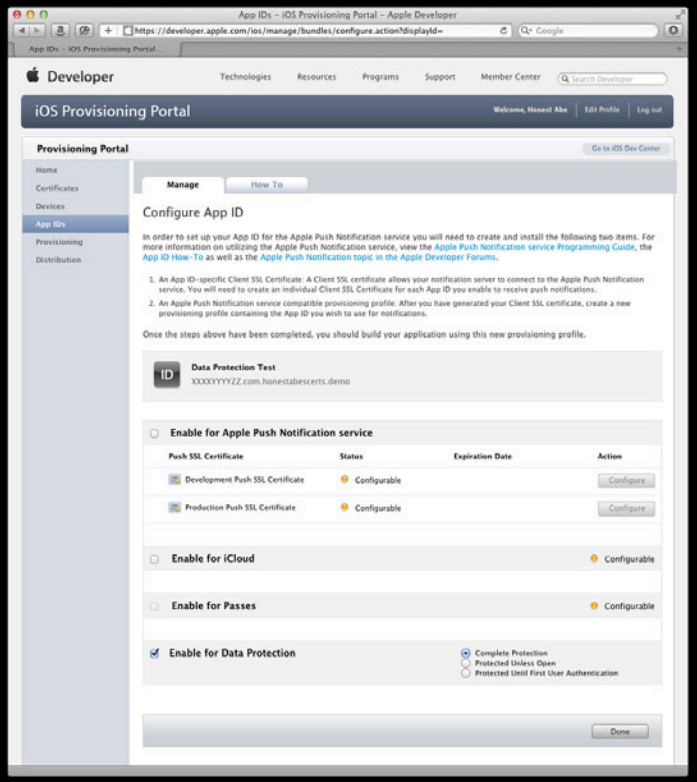
Data Protection for All Files

- Add `com.apple.developer.default-data-protection` entitlement
- Use `NSFileProtectionComplete` as its value
- Sets the default
  - Can still use APIs to control per file

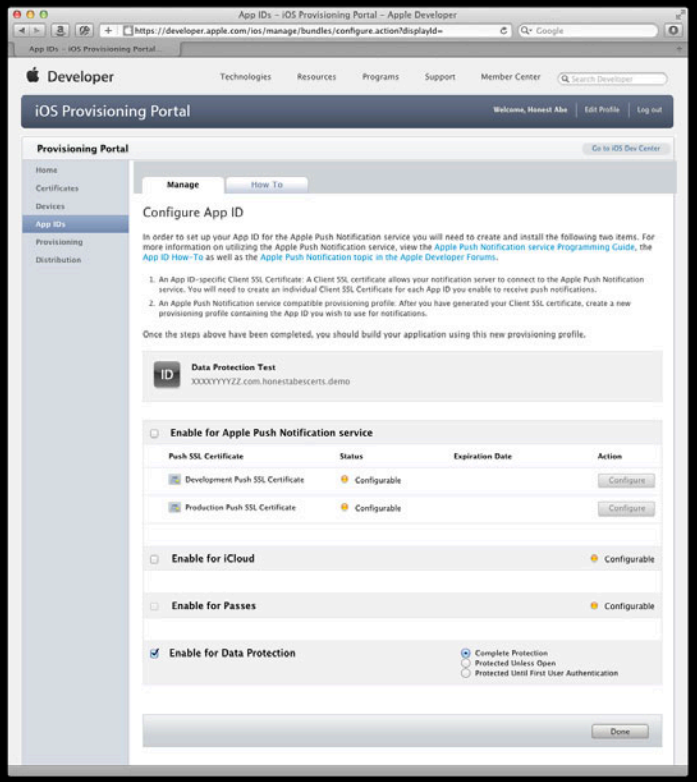
Data Protection for All Files

- Add `com.apple.developer.default-data-protection` entitlement
- Use `NSFileProtectionComplete` as its value
- Sets the default
  - Can still use APIs to control per file
- You'll need a new provisioning profile

Data Protection for All Files

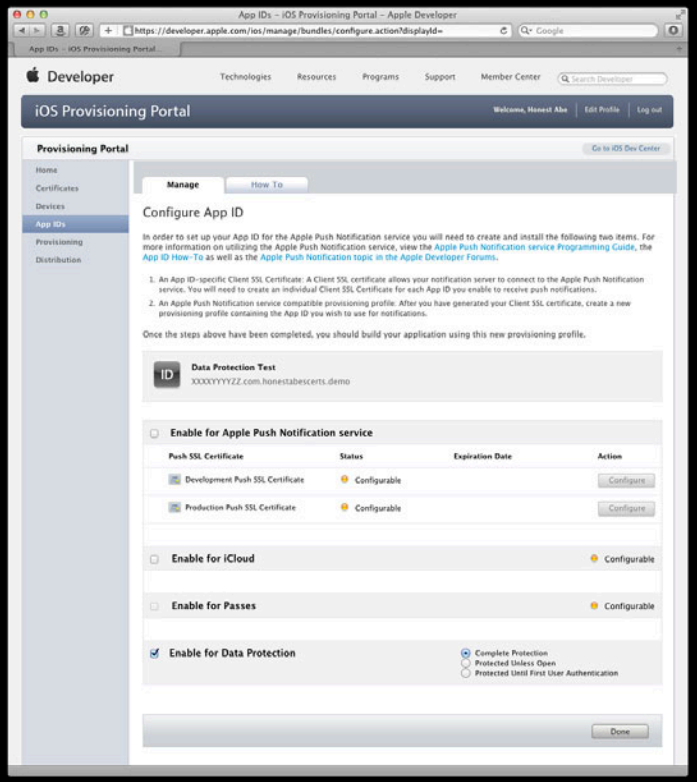


Data Protection for All Files



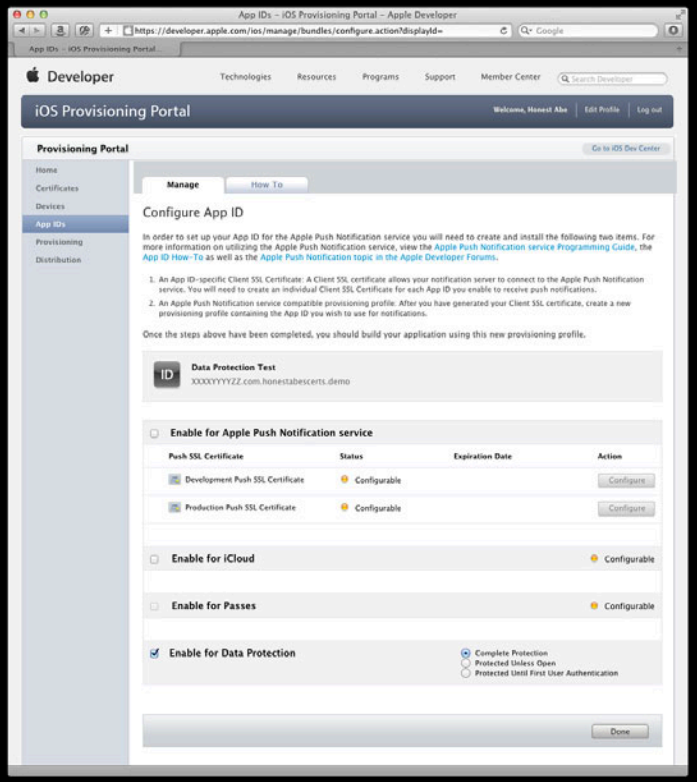
- Provisioning Portal

Data Protection for All Files



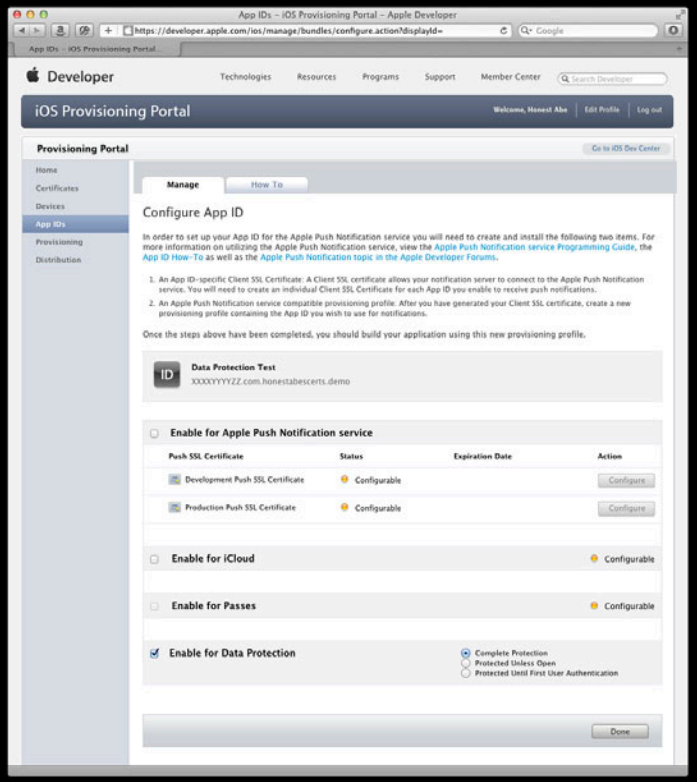
- Provisioning Portal
- App IDs

Data Protection for All Files



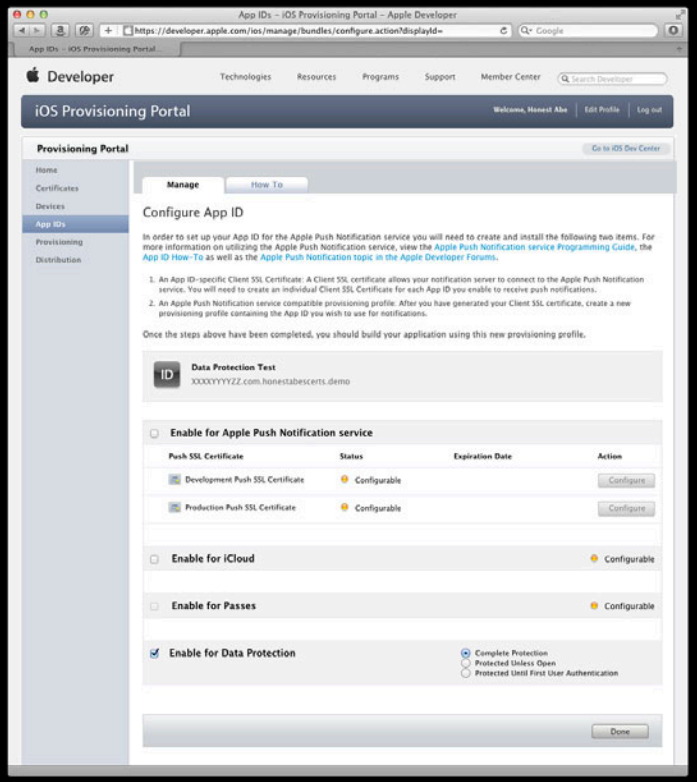
- Provisioning Portal
- App IDs
- “Enable for Data Protection”

Data Protection for All Files



- Provisioning Portal
- App IDs
- “Enable for Data Protection”
- Renew Provisioning Profile

Data Protection for All Files



- Provisioning Portal
- App IDs
- “Enable for Data Protection”
- Renew Provisioning Profile
- Download new profile
  or Renew in Xcode Organizer

Data Protection

Summary

Data Protection

Summary

Use `NSFileProtectionComplete`
unless you've a good reason not to

Keychain

Introduction

- What belongs in the Keychain
- Keychain Items are protected just like files
  - Migratability can be controlled

Keychain vs. Data Protection Classes

Availability	NSFileProtection	kSecAttrAccessible
When unlocked	…Complete	…WhenUnlocked
While locked	…CompleteUnlessOpen	N/A
After first unlock	…CompleteUntilFirstUserAuthentication	…AfterFirstUnlock
Always	…None	…Always

Keychain vs. Data Protection Classes

Availability	NSFileProtection	kSecAttrAccessible
When unlocked	…Complete	…WhenUnlocked
While locked	…CompleteUnlessOpen	N/A
After first unlock	…CompleteUntilFirstUserAuthentication	…AfterFirstUnlock
Always	…None	…Always

Keychain vs. Data Protection Classes

Availability	NSFileProtection	kSecAttrAccessible
When unlocked	…Complete	…WhenUnlocked
While locked	…CompleteUnlessOpen	N/A
After first unlock	…CompleteUntilFirstUserAuthentication	…AfterFirstUnlock
Always	…None	…Always

Keychain vs. Data Protection Classes

Availability	NSFileProtection	kSecAttrAccessible
When unlocked	…Complete	…WhenUnlocked
While locked	…CompleteUnlessOpen	N/A
After first unlock	…CompleteUntilFirstUserAuthentication	…AfterFirstUnlock
Always	…None	…Always

Keychain vs. Data Protection Classes

Availability	NSFileProtection	kSecAttrAccessible
When unlocked	…Complete	…WhenUnlocked
While locked	…CompleteUnlessOpen	N/A
After first unlock	…CompleteUntilFirstUserAuthentication	…AfterFirstUnlock
Always	…None	…Always

Keychain vs. Data Protection Classes

Availability	NSFileProtection	kSecAttrAccessible
When unlocked	…Complete	…WhenUnlocked
While locked	…CompleteUnlessOpen	N/A
After first unlock	…CompleteUntilFirstUserAuthentication	…AfterFirstUnlock
~~Always~~	~~…None~~	~~…Always~~

Non-Migrating Keychain Classes

Availability	NSFileProtection	kSecAttrAccessible
When unlocked	…Complete	…WhenUnlockedThisDeviceOnly
While locked	…CompleteUnlessOpen	N/A
After first unlock	…CompleteUntilFirstUserAuthentication	…AfterFirstUnlockThisDeviceOnly
~~Always~~	~~…None~~	~~…AlwaysThisDeviceOnly~~

Keychain

Item lookup

```
- (NSMutableDictionary *) queryForAccount:(NSString *)account {
    return [NSMutableDictionary dictionaryWithObjectsAndKeys:
        kSecClassGenericPassword, kSecClass,
        @"naivete", kSecAttrService, account, kSecAttrAccount, nil];
}
- (NSData *) passwordForAccount:(NSString *)account found:(BOOL *)found {
    NSMutableDictionary *query = [self queryForAccount: account];
         [query setObject: kCFBooleanTrue forKey: kSecReturnData];
    NSData *data = NULL;
         OSStatus status = SecItemCopyMatching(query, &data);
         *found = status != errSecItemNotFound;
         return data;
}
```

Keychain

Item lookup

```
- (NSMutableDictionary *) queryForAccount:(NSString *)account {
    return [NSMutableDictionary dictionaryWithObjectsAndKeys:
        kSecClassGenericPassword, kSecClass,
        @"naivete", kSecAttrService, account, kSecAttrAccount, nil];
}
- (NSData *) passwordForAccount:(NSString *)account found:(BOOL *)found {
    NSMutableDictionary *query = [self queryForAccount: account];
         [query setObject: kCFBooleanTrue forKey: kSecReturnData];
    NSData *data = NULL;
         OSStatus status = SecItemCopyMatching(query, &data);
         *found = status != errSecItemNotFound;
         return data;
}
```

Keychain

Item lookup

```
- (NSMutableDictionary *) queryForAccount:(NSString *)account {
    return [NSMutableDictionary dictionaryWithObjectsAndKeys:
        kSecClassGenericPassword, kSecClass,
        @"naivete", kSecAttrService, account, kSecAttrAccount, nil];
}
- (NSData *) passwordForAccount:(NSString *)account found:(BOOL *)found {
    NSMutableDictionary *query = [self queryForAccount: account];
        [query setObject: kCFBooleanTrue forKey: kSecReturnData];
    NSData *data = NULL;
        OSStatus status = SecItemCopyMatching(query, &data);
        *found = status != errSecItemNotFound;
        return data;
}
```

Keychain

Item lookup

```
- (NSMutableDictionary *) queryForAccount:(NSString *)account {
    return [NSMutableDictionary dictionaryWithObjectsAndKeys:
        kSecClassGenericPassword, kSecClass,
        @"naivete", kSecAttrService, account, kSecAttrAccount, nil];
}
- (NSData *) passwordForAccount:(NSString *)account found:(BOOL *)found {
    NSMutableDictionary *query = [self queryForAccount: account];
         [query setObject: kCFBooleanTrue forKey: kSecReturnData];
    NSData *data = NULL;
         OSStatus status = SecItemCopyMatching(query, &data);
         *found = status != errSecItemNotFound;
         return data;
}
```

Keychain

Item create

```
- (BOOL)setPassword: (NSData *)password forAccount:(NSString *)account {
    NSMutableDictionary *attrs = [self queryForAccount: account];
         [attrs setObject: password forKey: kSecValueData];
         [attrs setObject: kSecAttrAccessibleWhenUnlocked
             forKey: kSecAttrAccessible];
         OSStatus status = SecItemAdd(attrs, NULL);
         return status == noErr;
}
```

Keychain

Item create

```
- (BOOL)setPassword: (NSData *)password forAccount:(NSString *)account {
    NSMutableDictionary *attrs = [self queryForAccount: account];
        [attrs setObject: password forKey: kSecValueData];
        [attrs setObject: kSecAttrAccessibleWhenUnlocked
            forKey: kSecAttrAccessible];
        OSStatus status = SecItemAdd(attrs, NULL);
        return status == noErr;
}
```

Keychain

Item create

```
- (BOOL)setPassword: (NSData *)password forAccount:(NSString *)account {
    NSMutableDictionary *attrs = [self queryForAccount: account];
         [attrs setObject: password forKey: kSecValueData];
         [attrs setObject: kSecAttrAccessibleWhenUnlocked
             forKey: kSecAttrAccessible];
         OSStatus status = SecItemAdd(attrs, NULL);
         return status == noErr;
}
```

Keychain

Item update

```
- (BOOL)updatePassword: (NSData *)password forAccount:(NSString *)account {
    NSMutableDictionary *query = [self queryForAccount: account];
    NSMutableDictionary *attrs = [NSMutableDictionary dictionary];
        [attrs setObject: password forKey: kSecValueData];
        [attrs setObject: kSecAttrAccessibleWhenUnlocked
            forKey: kSecAttrAccessible];
        OSStatus status = SecItemUpdate(query, attrs);
        return status == noErr;
}
```

Keychain

Item update

```
- (BOOL)updatePassword: (NSData *)password forAccount:(NSString *)account {
    NSMutableDictionary *query = [self queryForAccount: account];
    NSMutableDictionary *attrs = [NSMutableDictionary dictionary];
        [attrs setObject: password forKey: kSecValueData];
        [attrs setObject: kSecAttrAccessibleWhenUnlocked
            forKey: kSecAttrAccessible];
        OSStatus status = SecItemUpdate(query, attrs);
        return status == noErr;
}
```

Keychain

Item update

```
- (BOOL)updatePassword: (NSData *)password forAccount:(NSString *)account {
    NSMutableDictionary *query = [self queryForAccount: account];
    NSMutableDictionary *attrs = [NSMutableDictionary dictionary];
        [attrs setObject: password forKey: kSecValueData];
        [attrs setObject: kSecAttrAccessibleWhenUnlocked
            forKey: kSecAttrAccessible];
        OSStatus status = SecItemUpdate(query, attrs);
        return status == noErr;
}
```

Keychain

Item update

```
- (BOOL)updatePassword: (NSData *)password forAccount:(NSString *)account {
    NSMutableDictionary *query = [self queryForAccount: account];
    NSMutableDictionary *attrs = [NSMutableDictionary dictionary];
        [attrs setObject: password forKey: kSecValueData];
        [attrs setObject: kSecAttrAccessibleWhenUnlocked
            forKey: kSecAttrAccessible];
        OSStatus status = SecItemUpdate(query, attrs);
        return status == noErr;
}
```

Keychain

High-level Keychain usage sample

```
-(BOOL) login: (NSString *) account {
    BOOL found = NO;
    NSData *pw = [self passwordForAccount: account found: &found];
    if ([self login: account password: pw]) return YES;
    pw = [self queryUserForPassword];
    if ([self login: account password: pw]) {
        if (found) [self updatePassword: pw forAccount: account];
        else [self setPassword: pw forAccount: account];
        return YES;
    }
    return NO;
}
```

Keychain

High-level Keychain usage sample

```
-(BOOL) login: (NSString *) account {
    BOOL found = NO;
    NSData *pw = [self passwordForAccount: account found: &found];
    if ([self login: account password: pw]) return YES;
    pw = [self queryUserForPassword];
    if ([self login: account password: pw]) {
        if (found) [self updatePassword: pw forAccount: account];
        else [self setPassword: pw forAccount: account];
        return YES;
    }
    return NO;
}
```

Keychain

High-level Keychain usage sample

```
-(BOOL) login: (NSString *) account {
    BOOL found = NO;
    NSData *pw = [self passwordForAccount: account found: &found];
    if ([self login: account password: pw]) return YES;
    pw = [self queryUserForPassword];
    if ([self login: account password: pw]) {
        if (found) [self updatePassword: pw forAccount: account];
        else [self setPassword: pw forAccount: account];
        return YES;
    }
    return NO;
}
```

Keychain

High-level Keychain usage sample

```
-(BOOL) login: (NSString *) account {
    BOOL found = NO;
    NSData *pw = [self passwordForAccount: account found: &found];
    if ([self login: account password: pw]) return YES;
    pw = [self queryUserForPassword];
    if ([self login: account password: pw]) {
        if (found) [self updatePassword: pw forAccount: account];
        else [self setPassword: pw forAccount: account];
        return YES;
    }
    return NO;
}
```

Keychain

High-level Keychain usage sample

```
-(BOOL) login: (NSString *) account {
    BOOL found = NO;
    NSData *pw = [self passwordForAccount: account found: &found];
    if ([self login: account password: pw]) return YES;
    pw = [self queryUserForPassword];
    if ([self login: account password: pw]) {
        if (found) [self updatePassword: pw forAccount: account];
        else [self setPassword: pw forAccount: account];
        return YES;
    }
    return NO;
}
```

Keychain

High-level Keychain usage sample

```
-(BOOL) login: (NSString *) account {
    BOOL found = NO;
    NSData *pw = [self passwordForAccount: account found: &found];
    if ([self login: account password: pw]) return YES;
    pw = [self queryUserForPassword];
    if ([self login: account password: pw]) {
        if (found) [self updatePassword: pw forAccount: account];
        else [self setPassword: pw forAccount: account];
        return YES;
    }
    return NO;
}
```

Demo

Andrew Whalley

Summary

Summary

- Protect your customers data

Summary

- Protect your customers data
  - Store secrets in the Keychain

Summary

- Protect your customers data
  - Store secrets in the Keychain
  - Use the entitlement/provisioning profile

Summary

- Protect your customers data
  - Store secrets in the Keychain
  - Use the entitlement/provisioning profile
  - Protect files with the best possible Data Protection class

Summary

- Protect your customers data
  - Store secrets in the Keychain
  - Use the entitlement/provisioning profile
  - Protect files with the best possible Data Protection class
  - Encrypt and authenticate network traffic

Summary

- Protect your customers data
  - Store secrets in the Keychain
  - Use the entitlement/provisioning profile
  - Protect files with the best possible Data Protection class
  - Encrypt and authenticate network traffic
- File Bugs!

Summary

- Protect your customers data
  - Store secrets in the Keychain
  - Use the entitlement/provisioning profile
  - Protect files with the best possible Data Protection class
  - Encrypt and authenticate network traffic
- File Bugs!
  - bugreport.apple.com

More Information

developer.apple.com

- Keychain Services Reference
- Certificate, Key, and Trust Services Reference
- NSFileManager Class Reference
- NSData Class Reference
- Core Data Framework Reference
- CFNetwork Framework Reference
- Secure Transport Reference

More Information

iOS Security

http://www.apple.com/iphone/business/resources/

Related Sessions

The OS X App Sandbox	Nob Hill Tuesday 10:15AM
Gatekeeper and Developer ID	Nob Hill Tuesday 11:30AM
The Security Framework	Nob Hill Tuesday 2:00 PM
Privacy Support in iOS and OS X	Pacific Heights Thursday 3:15PM
The OS X App Sandbox	Pacific Heights Friday 11:30AM

Source Code

- Search for
  - naivete AND wwdc
- Includes Oversharing from last year's session


WWDC2012