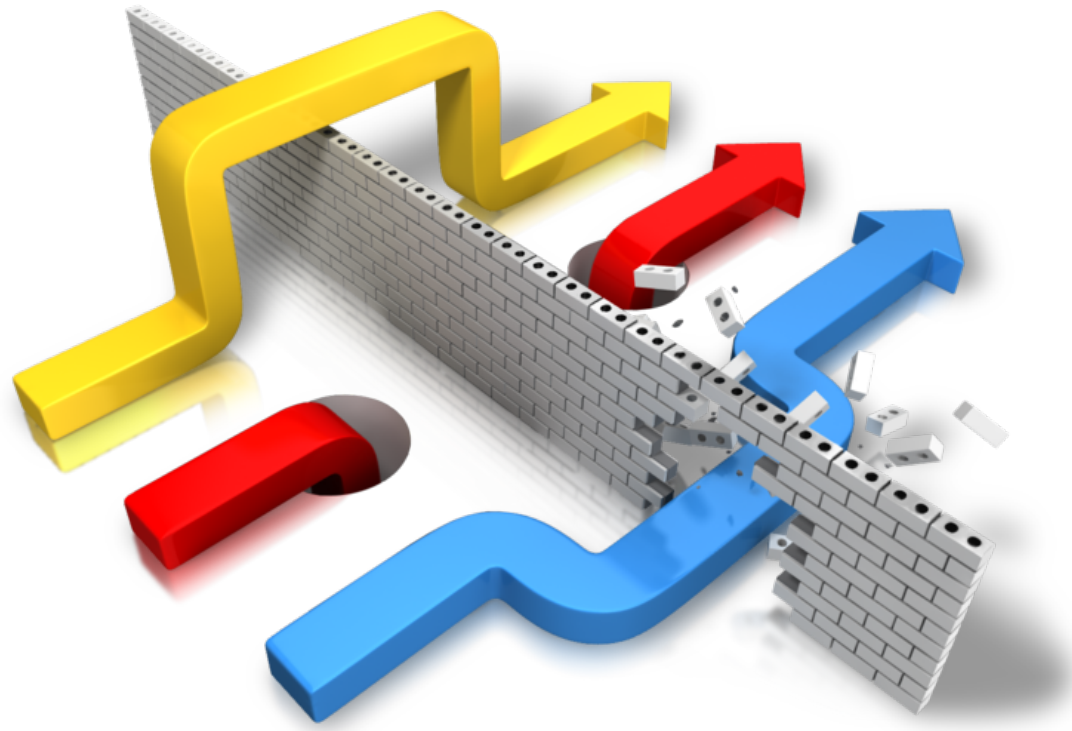


SECURE MOBILE APP DEVELOPMENT DOS & DON'TS

Rachid El Khayari – Testlab Mobile Security – Fraunhofer SIT



Security Engineering and Security Testing

Suppose a software product is developed without any functional testing at all. No alpha or beta testing. Write the code, compile it, and ship. The odds of this program working at all -- let alone being bug-free -- are zero. As the complexity of the product increases, so will the number of Bugs. Unfortunately, this is the current state of practice in security.

(Bruce Schneier)

Security engineering involves programming Satan's computer.

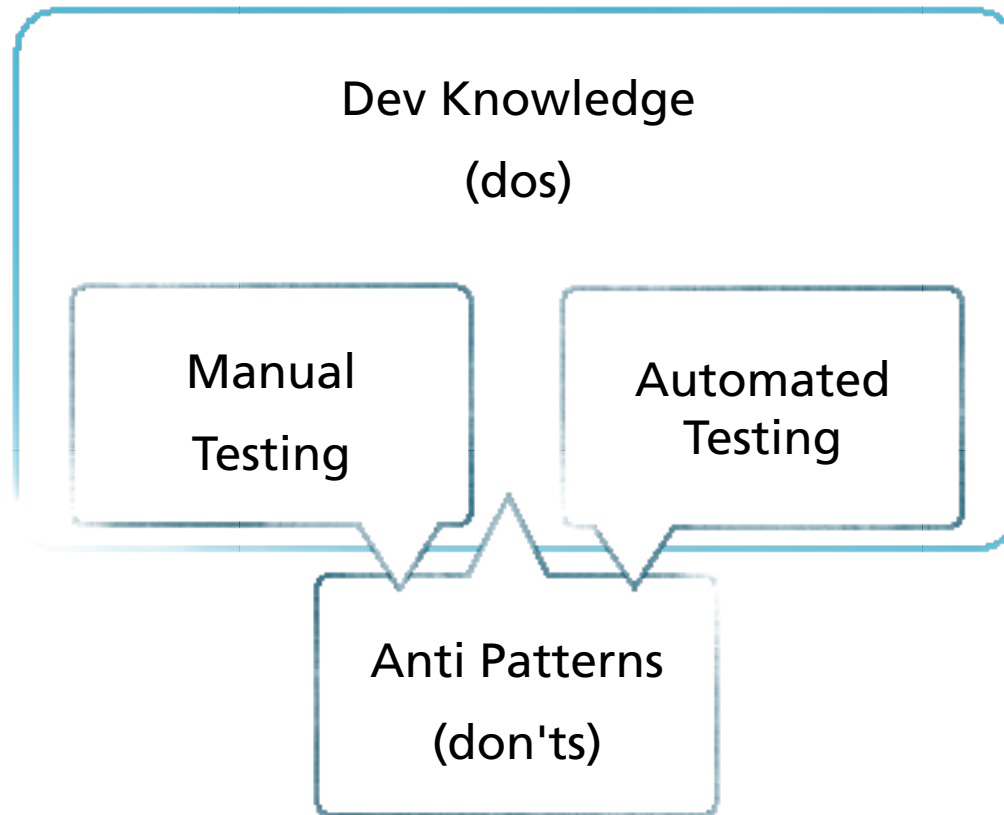
(Bruce Schneier acknowledging Ross Anderson)



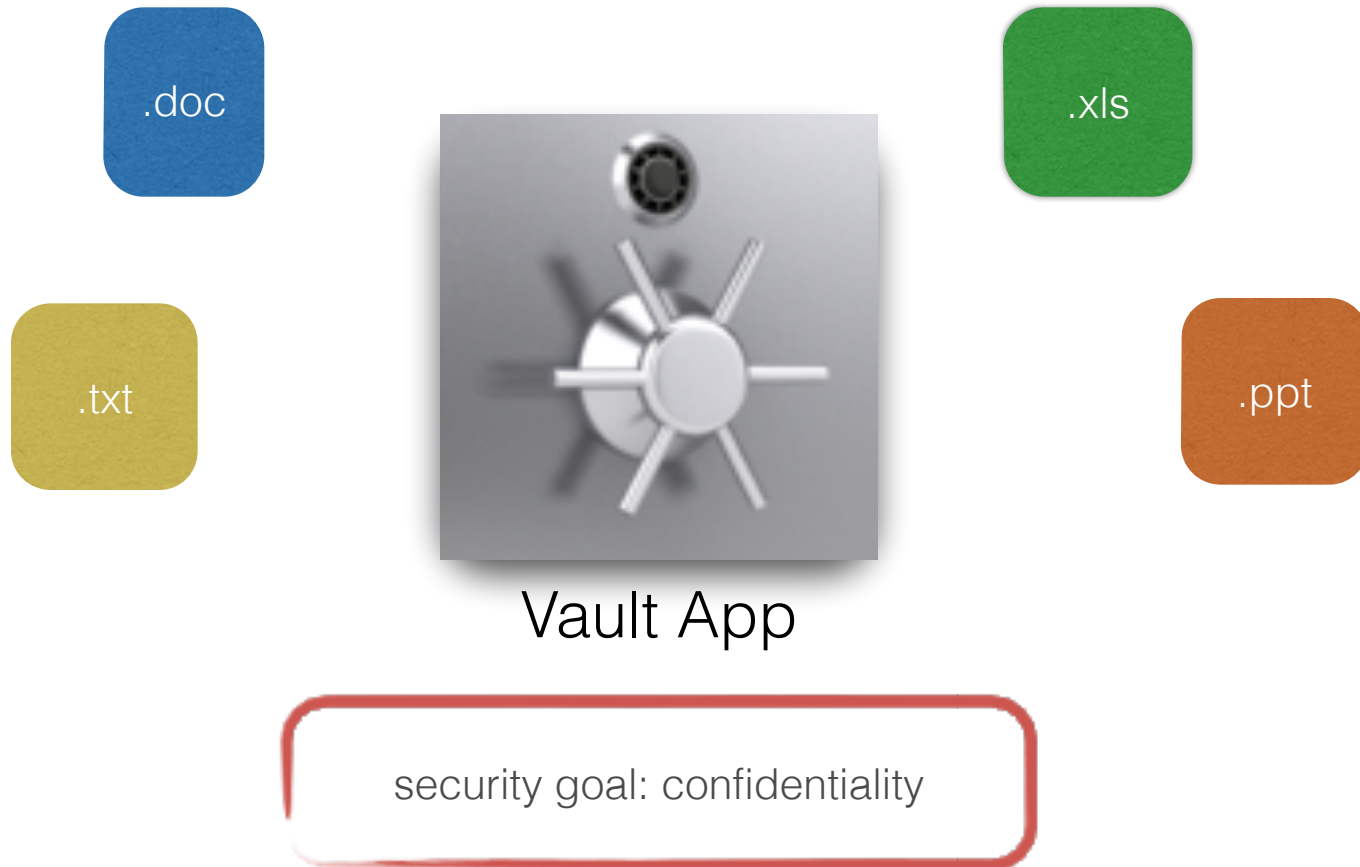
Where would you store your confidential data?



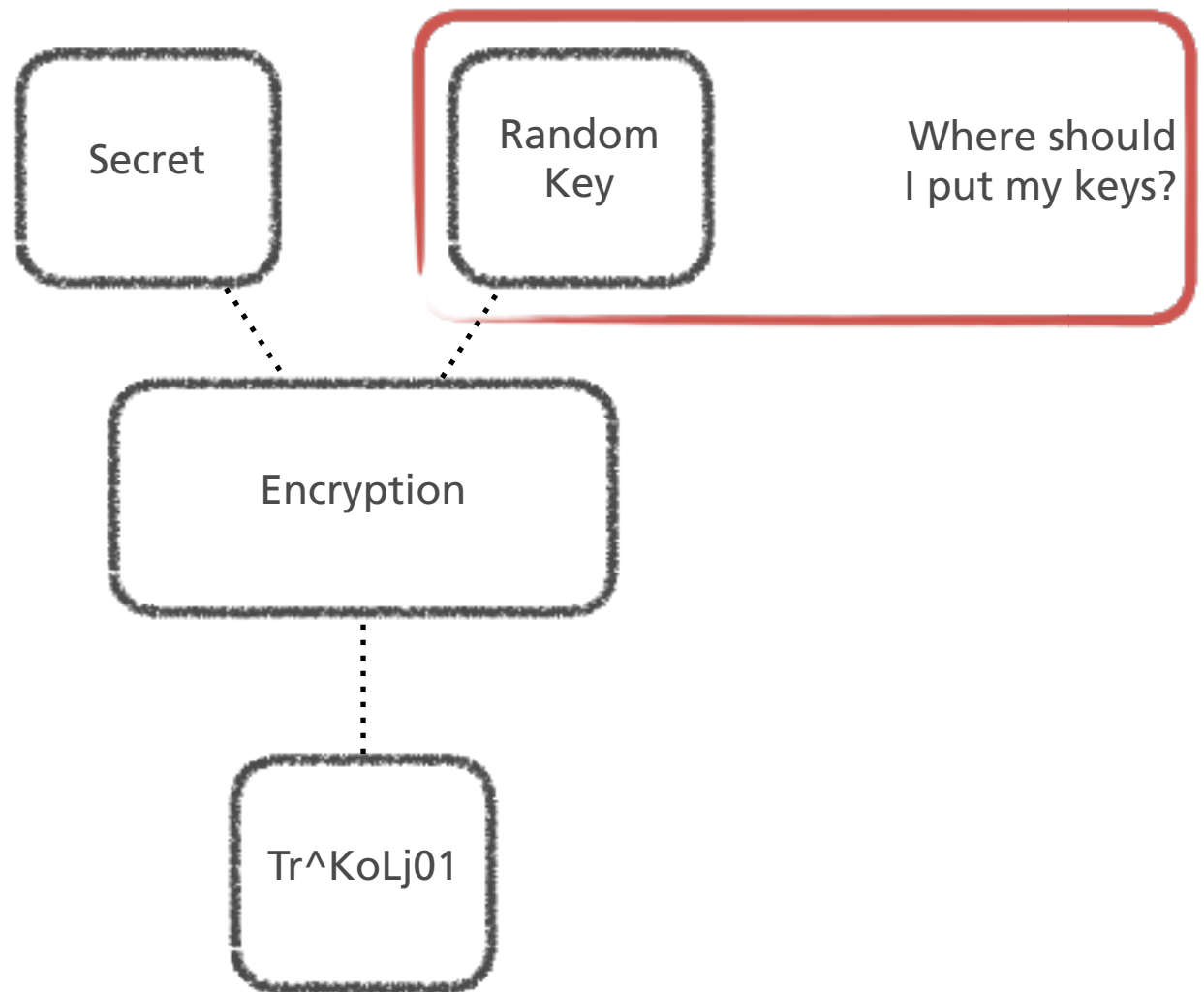
What do we know about (secure) development?



How (not) to develop a secure app!



Secrets! Where's the problem?



The lazy way to store secrets...



The screenshot shows a Stack Overflow page for the question "Storing secret keys on iPhone source and project resources". It features the Stack Overflow logo, navigation tabs for Questions, Tags, Users, Badges, and Unanswered, and sorting options for answers (active, oldest, votes). The top answer, by user kharrison, is highlighted with a red stamp that reads "don't do it!". The answer text discusses the security of storing secrets on a jail-broken iPhone and mentions the use of the keychain. The answer has 1,744 votes and was posted on April 4, 2010.

stackoverflow Questions Tags Users Badges Unanswered

Storing secret keys on iPhone source and project resources

2 Answers

active oldest votes

1

A lot depends on what you mean by secure. For normal device use it could be considered secure in that there is no way for a user to access it. However all bets are off for a jail-broken device which has complete access to the filesystem. So viewing a plist file in your application bundle is trivial on a jail-broken phone.

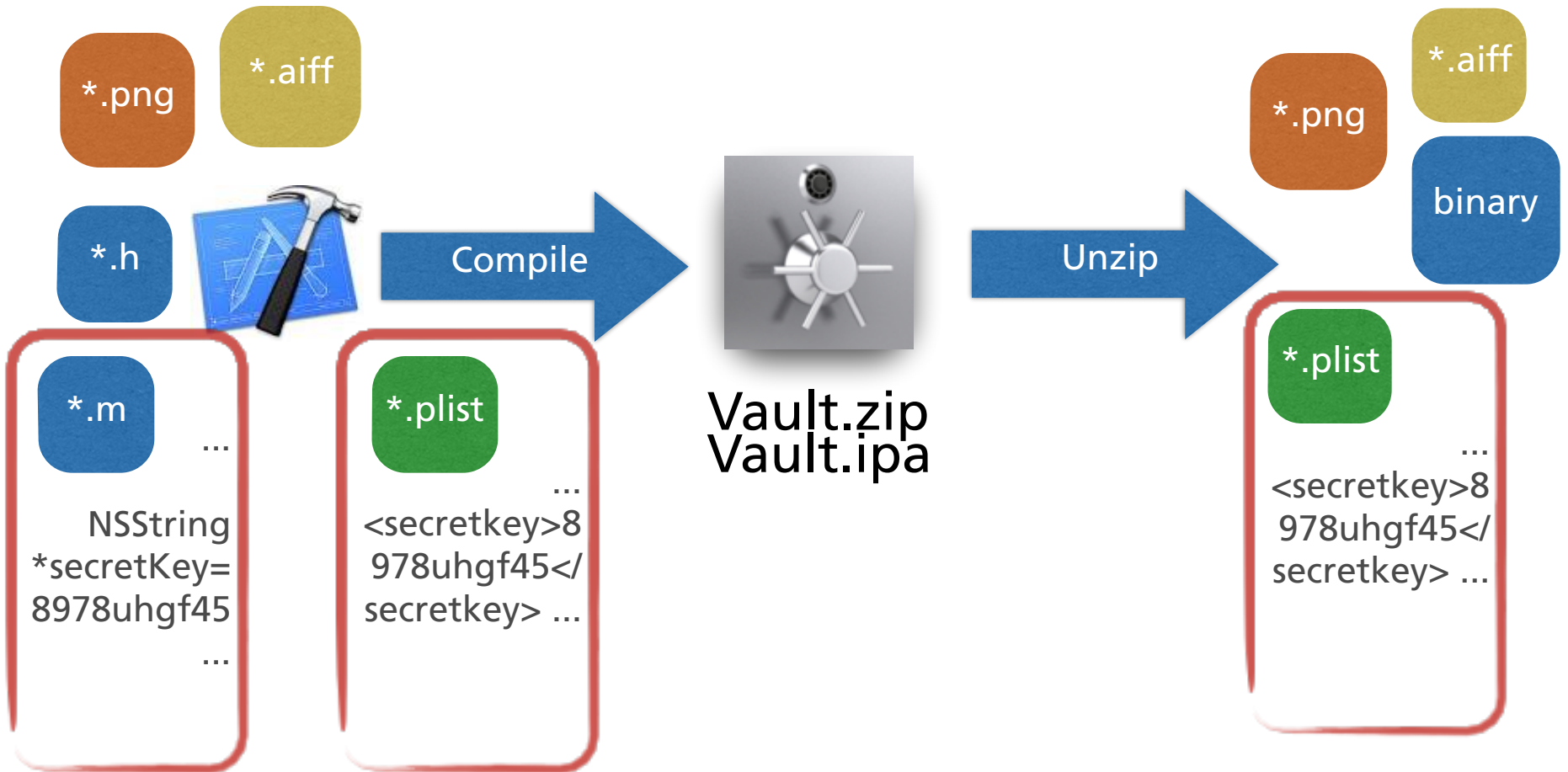
You might consider the use of the keychain which in theory would be safer and also has the advantage that the data will survive a reinstallation of your app. As before on a jail broken device nothing can be considered to be 100% secure but it depends how much trouble you want to go to.

share | improve this answer

answered Apr 4 '10 at 22:09

 kharrison 1,744 ●8●10

Hide and Seek!



Hide and Seek (II)



binary

```
> mv VaultApp.ipa VaultApp.zip  
> unzip VaultApp.zip  
> cd Payload/VaultApp/  
> strings VaultApp
```

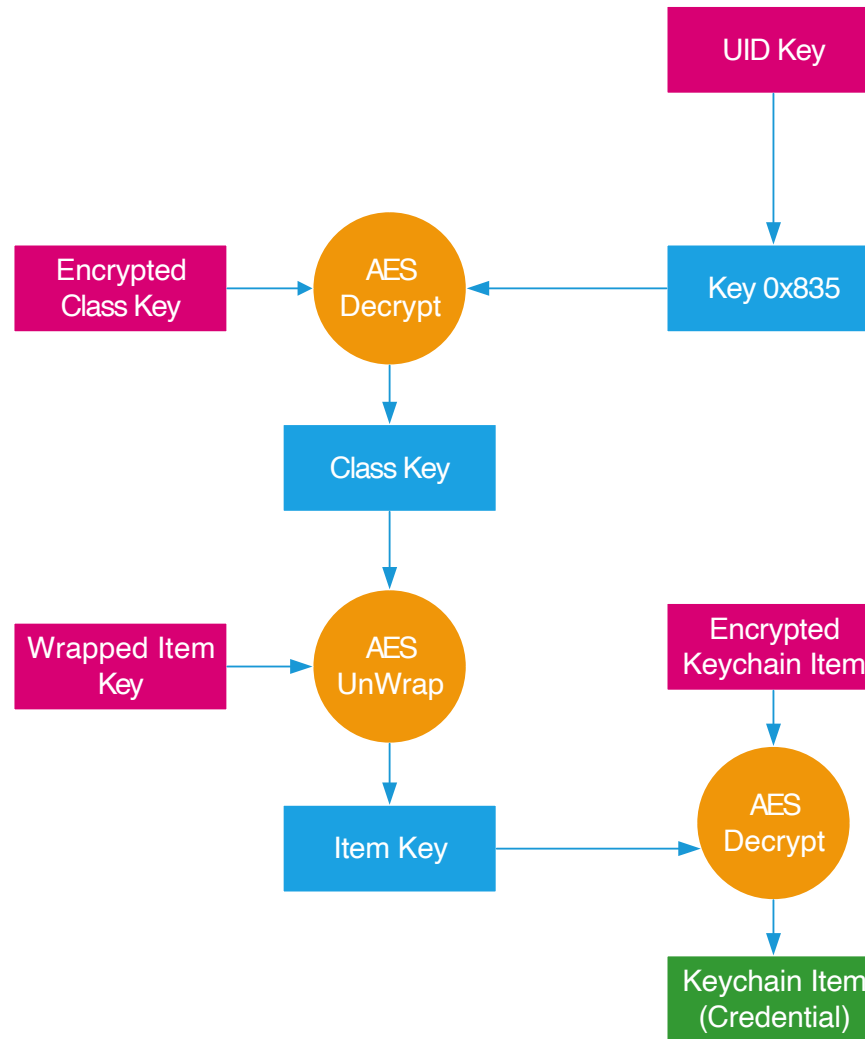
```
...  
NSString  
secretKey  
8978uhgf45  
...
```

iOS Keychain API

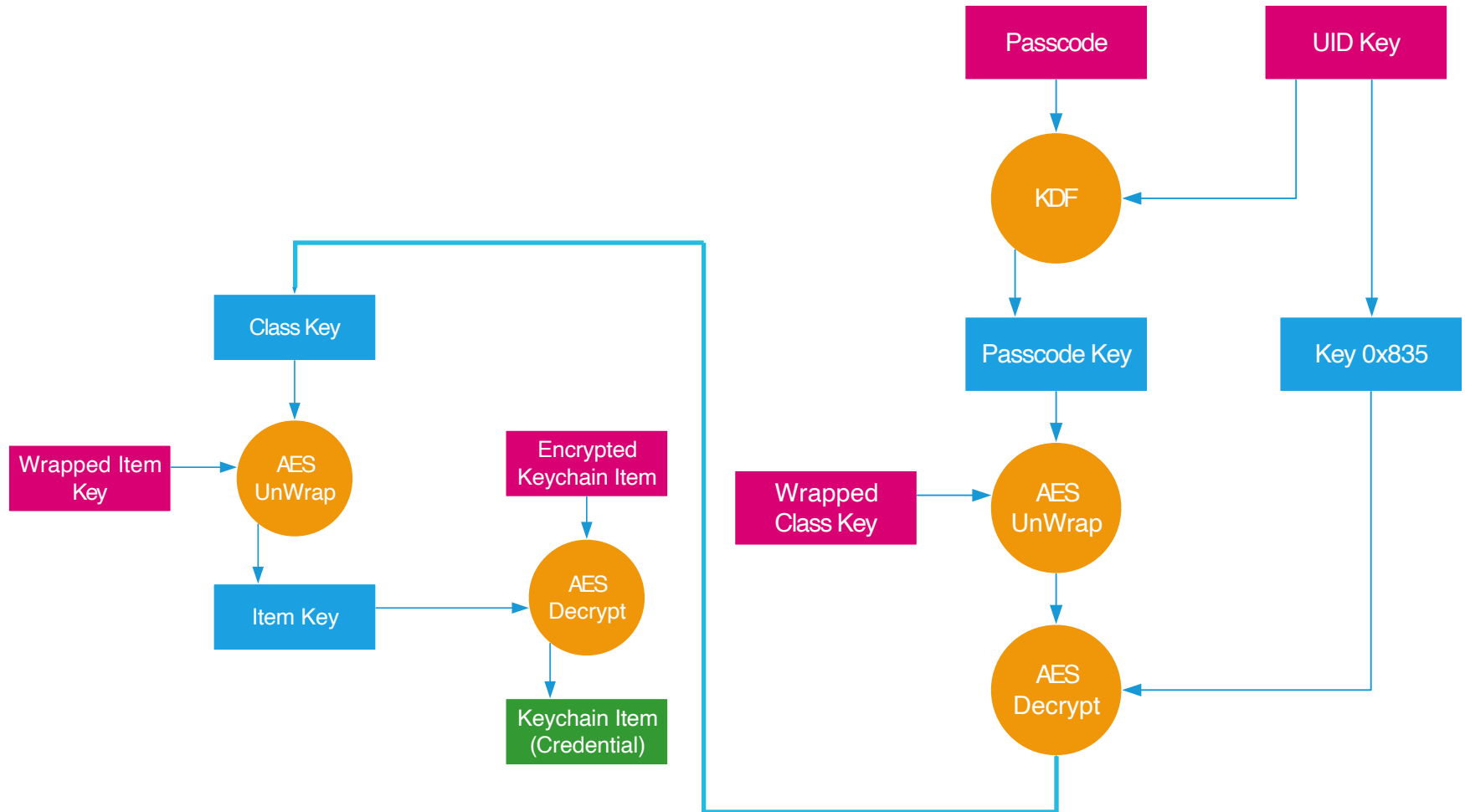
- Secure storage for secrets (e.g. passwords, keys, certs) on iOS
- Used by OS (e.g. Wifi-key, VPN-certs, Exchange-PW, ...)
- Might be used by 3rd-Party Apps through API

Keychain Class	Access when	Protected when
<i>WhenUnlocked</i>	<i>device is unlocked</i>	<i>device is locked</i>
<i>WhenUnlockedThisDeviceOnly</i>	<i>device is unlocked (bound to device)</i>	<i>device is locked (bound to device)</i>
<i>AfterFirstUnlock</i>	<i>after first unlock after reboot</i>	<i>before first unlock after reboot</i>
<i>AfterFirstUnlockThisDeviceOnly</i>	<i>after first unlock after reboot (bound to device)</i>	<i>before first unlock after reboot (bound to device)</i>
<i>Always</i>	<i>always</i>	<i>never</i>
<i>AlwaysThisDeviceOnly</i>	<i>always (bound to device)</i>	<i>never (bound to device)</i>

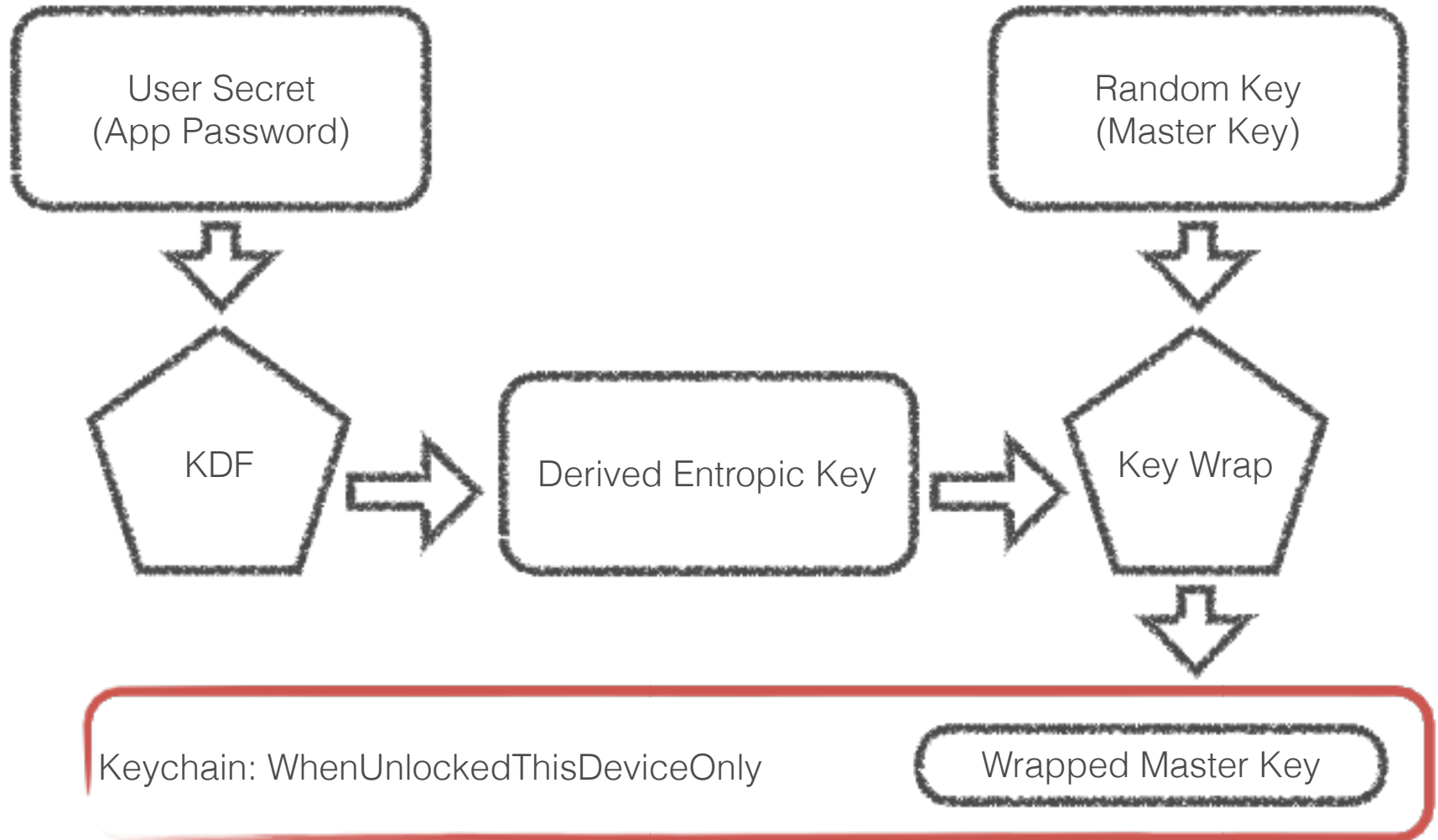
iOS Keychain without User Passcode



iOS Keychain with User Passcode



iOS User Secret Best Practice



Poor Man's Encryption

SUNDAY, FEBRUARY 15, 2009

Strong Encryption for Cocoa / Cocoa Touch

Please do not use this code! Instead, check out Jim Dovey's Common Crypto code from AQTToolkit.

don't do it!

AES is a strong encryption standard that has mostly replaced the aging DES standard. AES is widely used and fairly secure encryption mechanism (but I am not an expert at cryptography by any stretch of the imagination; I'm trusting experts for that opinion). AES supports three different key sizes, 128, 192, and 256 (the larger the key, the more secure the encryption and the more processing power it takes to encrypt or decrypt). Apple uses AES-128 and AES-256 in several places in Mac OS X, including for disk image encryption.

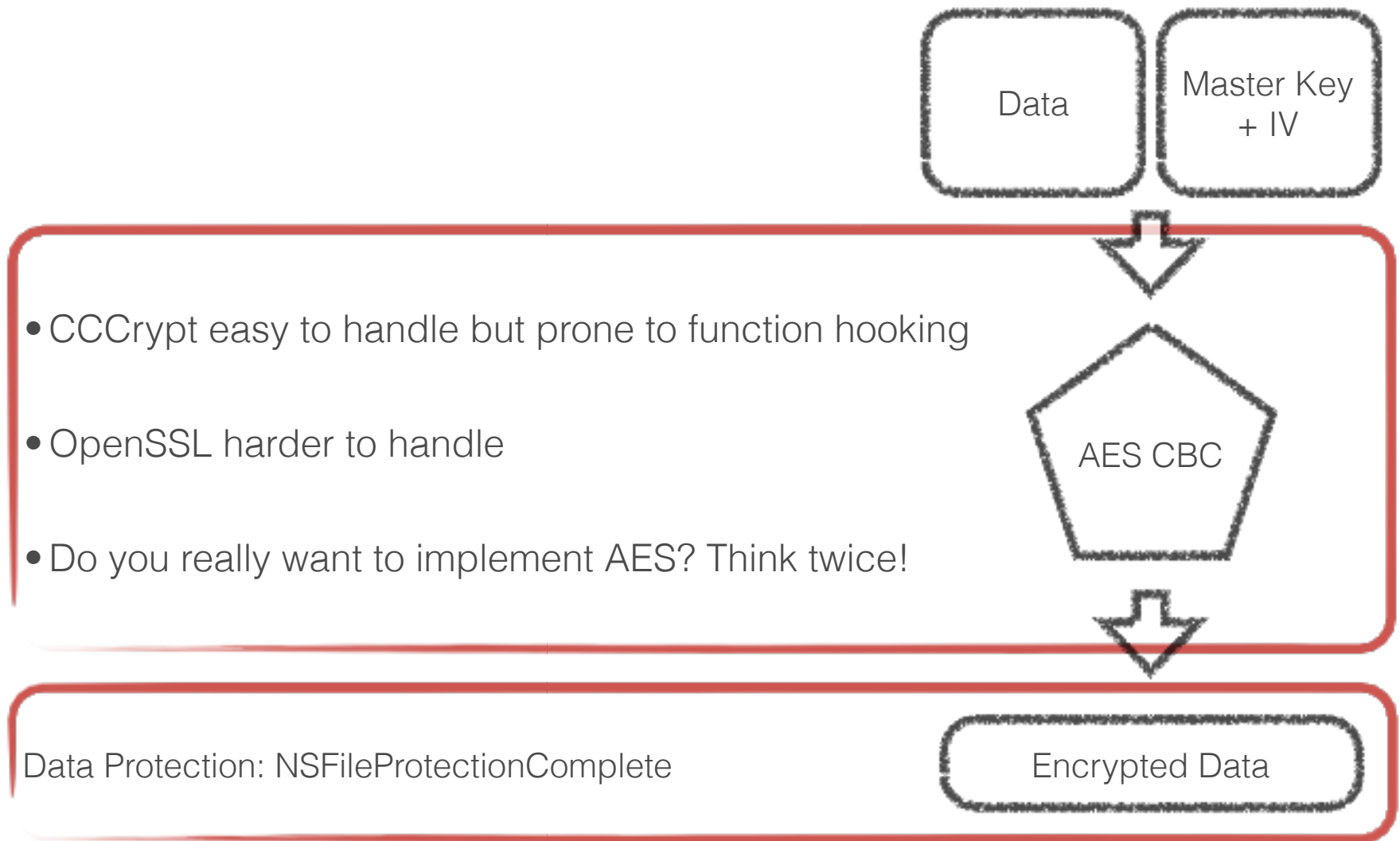
There are several public-domain implementations of AES. I chose a public domain implementation of AES by Philip J. Erdelsky to use as the basis some Objective-C categories that make encrypting and decrypting files and data using AES-256 easy.

iOS Data Protection API

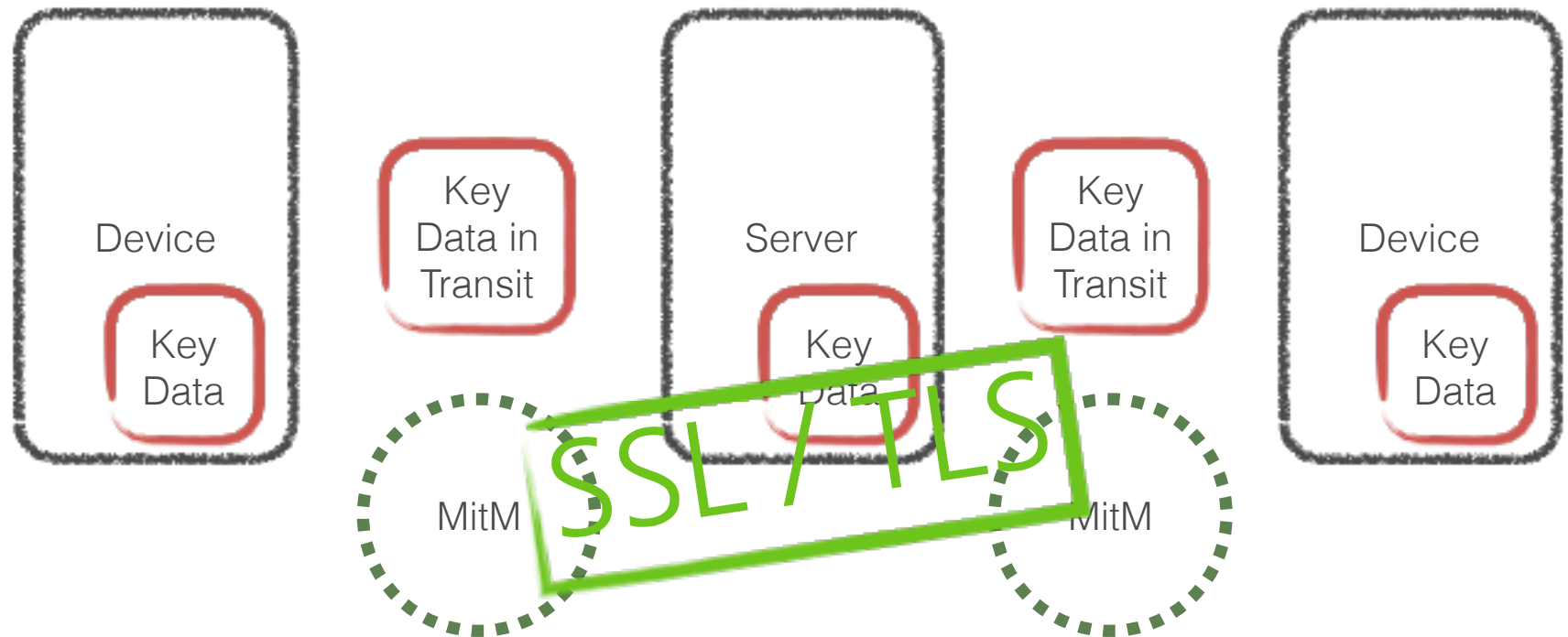
- File system and data encrypted with device keys (Dkey, EMF!)
- Data Protection API delivers additional encryption layer
- Might be used by 3rd-Party Apps through API

Data Protection Class	Access when	Protected when
<i>NSFileProtectionComplete</i>	<i>device is unlocked</i>	<i>device is locked</i>
<i>NSFileProtectionCompleteUnlessOpen</i>	<i>when unlocked or file was open at lock time</i>	<i>file was closed at lock time</i>
<i>NSFileProtectionCompleteUnlessFirstUserAuthentication</i>	<i>after first unlock after reboot</i>	<i>before first unlock after reboot</i>
<i>NSFileProtectionNone</i>	<i>always</i>	<i>never</i>

iOS Encryption Best Practice

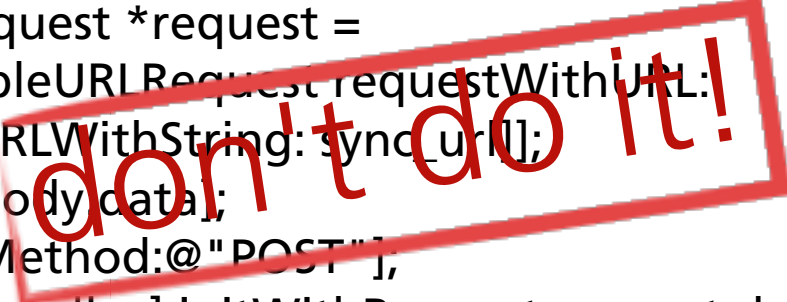


Data secured! Now I want to sync!



Transport security is easy


```
NSString *sync_url = @"http://example.com/sync/";
NSMutableURLRequest *request =
    [NSMutableURLRequest requestWithURL:
     [NSURL URLWithString: sync_url]];
[request setHTTPBody:data];
[request setHTTPMethod:@"POST"];
[[NSURLConnection alloc] initWithRequest:request delegate:self];
```



```
NSString *sync_url = @"https://example.com/sync/";
NSMutableURLRequest *request =
    [NSMutableURLRequest requestWithURL:
     [NSURL URLWithString: sync_url]];
[request setHTTPBody:data];
[request setHTTPMethod:@"POST"];
[[NSURLConnection alloc] initWithRequest:request delegate:self];
```

What about self signed certificates?

Apple Support Communities



Level 4 (1,320 points)

RickMaddy

🟢 This solved my question Re: iPhone SDK: letting user trust SSL certificates for NSURLConnection
Aug 7, 2008 12:49 AM (in response to digitalbeing)

I had the same problem. After some research I came across a solution that works but it's not officially supported. In your implementation class using NSURLRequest at the following lines:

```
@implementation NSURLRequest(DataController)
+ (BOOL)allowsAnyHTTPTSCertificateForHost:(NSString *)host
{
    return YES; // Or whatever logic
}
@end
```

This overrides this method. Simply returning YES here is potentially dangerous of course so you may need to return YES in a more controlled fashion. But it should get you further.

15" MacBook Pro Intel Core Duo 2.0GHz, 2GB RAM, Mac OS X (10.5.4), iPod touch 16GB, iPhone 16GB

👍 Like (0)

Contact

Rachid El Khayari
Rheinstr. 75
64295 Darmstadt

rachid.el.khayari@sit.fraunhofer.de

<http://www.sit.fraunhofer.de>

<http://sit4.me/tms>

<http://www.appicator.de>

