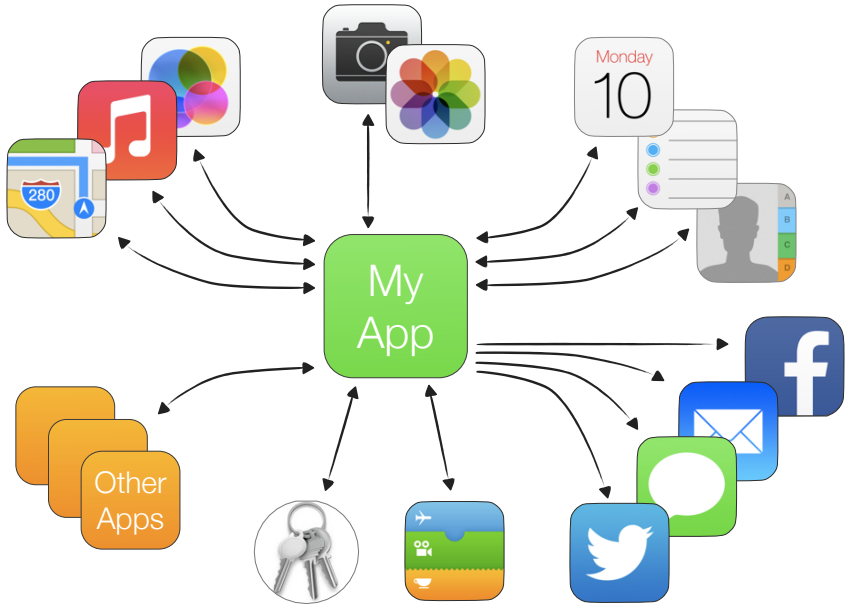
iPhone Application Programming
L12: Integration

Leonhard Lichtschlag
Media Computing Group
RWTH Aachen University

Winter Semester 2013/2014

http://hci.rwth-aachen.de/iphone

RWTH AACHEN UNIVERSITY



RWTH AACHEN UNIVERSITY

Accessing the Camera and Photos

RWTH AACHEN UNIVERSITY

Picking Photos

```
- (IBAction) pickImage:(id)sender
{
    if (![UIImagePickerController
isSourceTypeAvailable:UIImagePickerControllerSourceTypePhotoLibrary])
        return;

    // setup presenting view controller
    UIImagePickerController *cameraUI = [[UIImagePickerController alloc] init];
    cameraUI.allowsEditing = YES;
    cameraUI.delegate = self;

    cameraUI.sourceType = UIImagePickerControllerSourceTypePhotoLibrary;
    cameraUI.mediaTypes = [UIImagePickerController

availableMediaTypesForSourceType:UIImagePickerControllerSourceTypePhotoLibrary];

    [self presentViewController:cameraUI animated:YES completion:nil];
}
```


RWTH AACHEN UNIVERSITY

Picking Photos

```
- (void) imagePickerController:(UIImagePickerController *)picker
         didFinishPickingMediaWithInfo:(NSDictionary *)info
{
    // figure out which image to use
    UIImage *originalImage, *editedImage, *imageToSave;
    editedImage = (UIImage *) info[UIImagePickerControllerEditedImage];
    originalImage = (UIImage *) info[UIImagePickerControllerOriginalImage];
    if (editedImage)
        imageToSave = editedImage;
    else
        imageToSave = originalImage;

    self.imageView.image = imageToSave;
    [self dismissViewControllerAnimated:YES completion:nil];
}
```

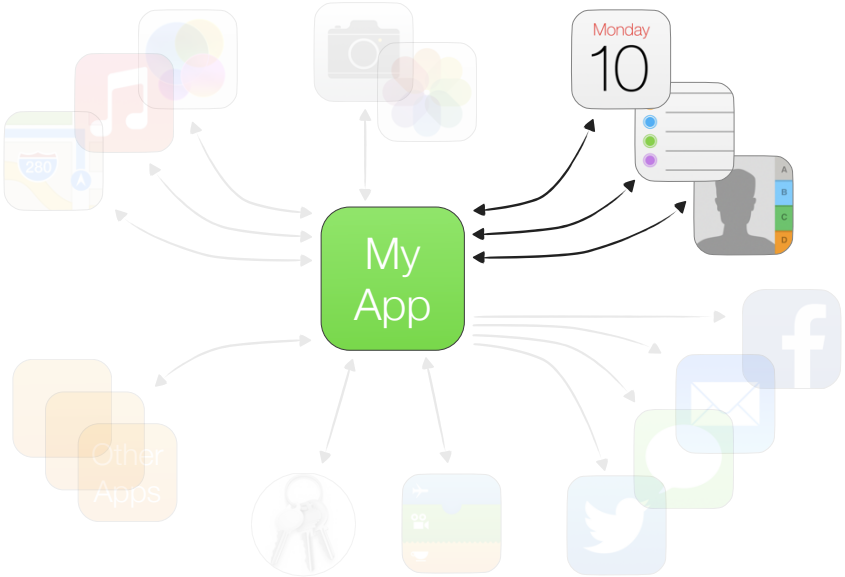
RWTH AACHEN UNIVERSITY

UIImagePicker

- permission dialogue
- the look of the camera controls can customized
- AVFoundation for detailed camera controls
- AVAsset for detailed image information

RWTH AACHEN UNIVERSITY

Accessing the Organizer Apps



RWTH AACHEN UNIVERSITY

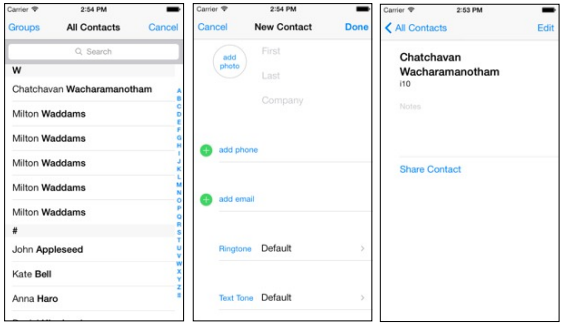
Using the Address Book data

- AddressBookUI Framework
  - Provides the UI elements from the Address Book
  - Objective-C framework
  - Using existing contacts
  - Adding new contacts
- AddressBook Framework
  - C-framework (Core Foundation)
  - full library access

RWTH AACHEN UNIVERSITY

Address Book UI

- ABPeoplePickerNavigationController
- ABUnknownPersonViewController
- ABNewPersonViewController
- ABPersonViewController



RWTH AACHEN UNIVERSITY

Demo

RWTH AACHEN UNIVERSITY

Creating a Contact (in Code)

```
// create a new contact
ABRecordRef newPerson = ABPersonCreate();
ABRecordSetValue(newPerson, kABPersonFirstNameProperty, CFSTR("Milton"), NULL);
ABRecordSetValue(newPerson, kABPersonLastNameProperty, CFSTR("Waddams"), NULL);
ABRecordSetValue(newPerson, kABPersonOrganizationProperty, CFSTR("Initech"),
NULL);

// Show the contact in an address book UI
ABPersonViewController *pvc = [[ABPersonViewController alloc] init];
pvc.allowsEditing = YES;
pvc.displayedPerson = newPerson;
[self.navigationController pushViewController:pvc animated:YES];
[pvc release];

// Add the contact to the address book
ABAddressBookRef addressBook = ABAddressBookCreate();
ABAddressBookAddRecord(addressBook, newPerson, &error);
if (error == NULL)
{
   ABAddressBookSave(addressBook, &error);
}
CFRelease(newPerson);
CFRelease(addressBook);
```


RWTH AACHEN UNIVERSITY

Creating a Contact (in Code)

```
// create a new contact
ABRecordRef newPerson = ABPersonCreate();
ABRecordSetValue(newPerson, kABPersonFirstNameProperty, CFSTR("Milton"), NULL);
ABRecordSetValue(newPerson, kABPersonLastNameProperty, CFSTR("Waddams"), NULL);
ABRecordSetValue(newPerson, kABPersonOrganizationProperty, CFSTR("Initech"),
NULL);

// Show the contact in an address book UI
ABPersonViewController *pvc = [[ABPersonViewController alloc] init];
pvc.allowsEditing = YES;
pvc.displayedPerson = newPerson;
[self.navigationController pushViewController:pvc animated:YES];
[pvc release];

// Add the contact to the address book
ABAddressBookRef addressBook = ABAddressBookCreate();
ABAddressBookAddRecord(addressBook, newPerson, &error);
if (error == NULL)
{
   ABAddressBookSave(addressBook, &error);
}
CFRelease(newPerson);
CFRelease(addressBook);
```


RWTH AACHEN UNIVERSITY

Picking a Contact

```
// Show a PeoplePicker and return the result to self as a delegate
- (IBAction) chooseContact;
{
   ABPeoplePickerNavigationController *picker =
                          [[ABPeoplePickerNavigationController alloc] init];
   picker.peoplePickerDelegate = self;
   [self presentModalViewController:picker animated:YES];
}


// This method gets called when the user picks a contact from the list
- (BOOL) peoplePickerNavigationController:(ABPeoplePickerNavigationController
*)pnc
       shouldContinueAfterSelectingPerson:(ABRecordRef)person
{
  // Get the data out of the c struct
   CFStringRef retainedName =
ABRecordCopyValue(person,kABPersonLastNameProperty);

   NSString* ARCname        = CFBridgingRelease(retainedName);
   self.firstNameLabel.text = name;

   // Remove view, and do not continue picking on people
   [self dismissModalViewControllerAnimated:YES];
   return NO;
}
```


Picking a Contact

```
// Show a PeoplePicker and return the result to self as a delegate
- (IBAction) chooseContact;
{
   ABPeoplePickerNavigationController *picker =
                          [[ABPeoplePickerNavigationController alloc] init];
   picker.peoplePickerDelegate = self;
   [self presentModalViewController:picker animated:YES];
}


// This method gets called when the user picks a contact from the list
- (BOOL) peoplePickerNavigationController:(ABPeoplePickerNavigationController
*)pnc
       shouldContinueAfterSelectingPerson:(ABRecordRef)person
{
  // Get the data out of the c struct
   CFStringRef retainedName =
ABRecordCopyValue(person,kABPersonLastNameProperty);

   NSString* ARCname        = CFBridgingRelease(retainedName);
   self.firstNameLabel.text = name;

   // Remove view, and do not continue picking on people
   [self dismissModalViewControllerAnimated:YES];
   return NO;
}
```


Getting authorization from the

- Permission dialogue shows up automatically when an API needs it
- Info.plist entries for privacy statements
- Use the API as late as possible
  - close to the relevant interaction
- This also applies for Photos, Events, Reminders, ...

Working with the Address Book C-

- ABAddressBookRef
  - Created with ABAddressBookCreateWithOptions
  - Multiple instances
  - One database
  - Make sure that an instance is only used by one thread

Groups

- Group of contacts
- Group records have only one property
  - kABGroupNameProperty
- Working on groups
  - ABGroupAddMember
  - ABGroupRemoveMember
  - ABGroupCopyArrayOfAllMembers
  - ABGroupCopyArrayOfAllMembersWithSortOrdering

EventKit

Initializing an Event Store

- Access the event database with an EKEventStore
- Long initialization and release times
- Use one event store repeatedly
- Load it when app launches
- Ask permission

```
@property (nonatomic, retain) EKEventStore *eventStore;
self.eventStore = [[EKEventStore alloc] init];
[self.eventStore = requestAccessToEntityType:EKEntityTypeReminder
                                  completion:nil];
```


Fetching Events

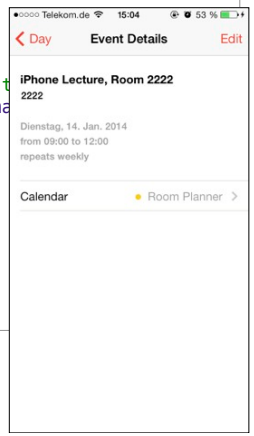
```
- (NSArray *) fetchEventsForToday
{
 // Do the proper math for a time span of
24 hours into the future
 NSDate *startDate = [[NSDate alloc] init];
 NSCalendar *cal = [[NSCalendar alloc]


initWithCalendarIdentifier:NSGregorianCalen
dar];
 NSDateComponents *offsetComponents =
[[NSDateComponents alloc] init];
```

EKEventViewController

```
// Show details for a selected event in a table view
EKEventViewController* detailVC =
      [[EKEventViewController alloc] initWithNibName:nil
                                              bundle:nil];
detailVC.event = [self.eventsList objectAtIndex:indexPath.row];

// Allow event editing
detailVC.allowsEditing = YES;

// Push the view controller on the navigationController's stack
[self.navigationController pushViewController:detailVC animated:YES];
```


EKEventViewController

```
// Show details for a selected event in a table view
EKEventViewController* detailVC =
      [[EKEventViewController alloc] initWithNibName:nil
                                              bundle:nil];
detailVC.event = [self.eventsList objectAtIndex:indexPath.row];

// Allow event editing
detailVC.allowsEditing = YES;

// Push the view controller on the navigationController's st
[self.navigationController pushViewController:detailVC anima
```

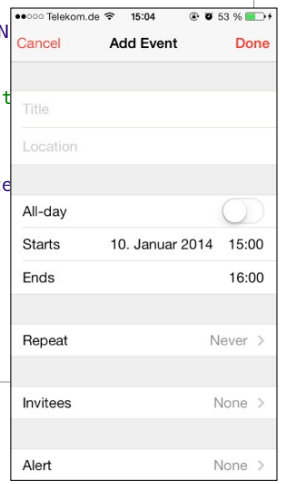


EKEventEditViewController

```
- (void) addEvent:(id)sender
{
   // Create an EditViewController
   EKEventEditViewController *editController =
         [[EKEventEditViewController alloc] initWithNibName:nil bundle:nil];
   editController.editViewDelegate = self;

   // set the editController's event store to the current event store
   editController.eventStore = self.eventStore;

   // show the view controller
   [self presentModalViewController:addController animated:YES];
   [editController release];
}
```


EKEventEditViewController

```
- (void) addEvent:(id)sender
{
   // Create an EditViewController
   EKEventEditViewController *editController =
         [[EKEventEditViewController alloc] initWithNibN
   editController.editViewDelegate = self;

   // set the editController's event store to the current
   editController.eventStore = self.eventStore;

   // show the view controller
   [self presentModalViewController:addController animate
   [editController release];
}
```



EKEventEditViewDelegate

```
- (void) eventEditViewController:(EKEventEditViewController *)controller
          didCompleteWithAction:(EKEventEditViewAction)action
{
   switch (action)
   {
      case EKEventEditViewActionSaved: // A new event was created. Add it
         [controller.eventStore saveEvent:controller.event
                                     span:EKSpanThisEvent error:&error];
         break;

      case EKEventEditViewActionDeleted: // The event was deleted
         [controller.eventStore removeEvent:thisEvent
                                       span:EKSpanThisEvent error:&error];

      case EKEventEditViewActionCanceled: // Edit action canceled, do
nothing.
      default:
         break;
   }
   // Dismiss the modal view controller
   [controller dismissModalViewControllerAnimated:YES];
}
```


Create Recurring Events

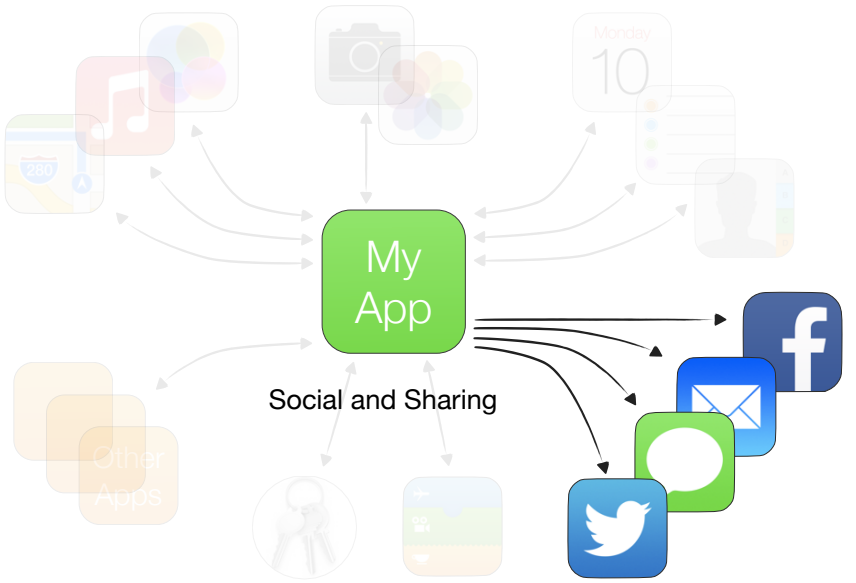
```
// Create a recurrence rule and assign it to the event
EKRecurrenceRule* recRule=
      [[EKRecurrenceRule alloc]
            initRecurrenceWithFrequency:EKRecurrenceFrequencyWeekly
               interval:1
                  end:[EKRecurrenceEnd recurrenceEndWithOccurrenceCount:8]];

newEvent.recurrenceRule = recRule;
```

```
   initRecurrenceWithFrequency:(EKRecurrenceFrequency)type
                      interval:(NSInteger)interval
                 daysOfTheWeek:(NSArray *)days
                daysOfTheMonth:(NSArray *)monthDays
               monthsOfTheYear:(NSArray *)months
                weeksOfTheYear:(NSArray *)weeksOfTheYear
                 daysOfTheYear:(NSArray *)daysOfTheYear
                  setPositions:(NSArray *)setPositions
                           end:(EKRecurrenceEnd *)end];
```


Observing Events

- Event updated in the background
  - iCloud sync
  - Exchange sync
- Check if UI update necessary

```
[[NSNotificationCenter defaultCenter] addObserver:self
                                         selector:@selector(storeChanged:)
                                             name:EKEventStoreChangedNotification
                                           object:eventStore];
```

MessageUI

- Create emails and texts within your application
- Your application is not quit
- Available in iOS 5 and higher
- iOS6 integrates Facebook and Twitter

Composing an Email

```
// Create a Mail Compose View Controller
// First, check if the device is configured to send mail
if ([MFMailComposeViewController canSendMail] == YES)
{
   // get a new view
   MFMailComposeViewController *composeViewController =
                                [[MFMailComposeViewController alloc] init];
   composeViewController.mailComposeDelegate = self;

   // Set initial values
   [composeViewController setToRecipients:
                [NSArray arrayWithObject:toTextField.text]];
   [composeViewController setSubject:subjectTextField.text];

   // Bring it to the screen
   [self presentModalViewController:composeViewController animated:YES];
   [composeViewController release];
}
```


Mail Composer Delegate

```
- (void) mailComposeController:(MFMailComposeViewController*)controller
          didFinishWithResult:(MFMailComposeResult)result error:(NSError*)error
{
    // Notifies users about errors associated with the interface
    switch (result)
    {
        case MFMailComposeResultCancelled:
            messageLabel.text = @"Result: canceled";
            break;
        case MFMailComposeResultSaved:
            messageLabel.text = @"Result: saved";
            break;
        case MFMailComposeResultSent:
            messageLabel.text = @"Result: sent";
            break;
        case MFMailComposeResultFailed:
            messageLabel.text = @"Result: failed";
            break;
        default:
            messageLabel.text = @"Result: not sent";
            break;
    }
    [self dismissModalViewControllerAnimated:YES];
}
```


Send a Message (SMS)

```
- (void) displaySMSComposer
{
    if (![MFMessageComposeViewController canSendText])
       return;

  // get a new view
   MFMessageComposeViewController *composer =
                        [[MFMessageComposeViewController alloc] init];
   composer.messageComposeDelegate = self;


    // configure
    [composer setRecipients: [NSArray arrayWithObject:toTextField.text]];
    [composer setBody:@"This is the iOS class!"];


    // show on screen
   [self presentModalViewController:picker animated:YES];
   [composer release];
}
```

Message Composer Delegate

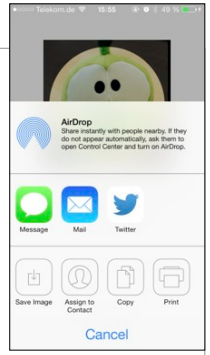
```
- (void) messageComposeViewController:(MFMessageComposeViewController
*)control
                 didFinishWithResult:(MessageComposeResult)result
{
  // Notifies users about errors associated with the interface
  switch (result)
  {
      case MessageComposeResultCancelled:
         feedbackMsg.text = @"Result: SMS sending canceled";
         break;
      case MessageComposeResultSent:
         feedbackMsg.text = @"Result: SMS sent";
         break;
      case MessageComposeResultFailed:
         feedbackMsg.text = @"Result: SMS sending failed";
         break;
      default:
         feedbackMsg.text = @"Result: SMS not sent";
         break;
  }
  [self dismissModalViewControllerAnimated:YES];
}
```


Social Framework

```
- (void) shareWithFacebook
{
    NSString    *textToShare    = self.textField.text;
    UIImage     *imageToShare   = self.imageView.image;

    SLComposeViewController *facebookPostVC = [SLComposeViewController
              composeViewControllerForServiceType:SLServiceTypeFacebook];

    [facebookPostVC setInitialText:textToShare];
    [facebookPostVC addImage:imageToShare];

    [self presentViewController:facebookPostVC
                       animated:YES
                       completion:nil];
}
```

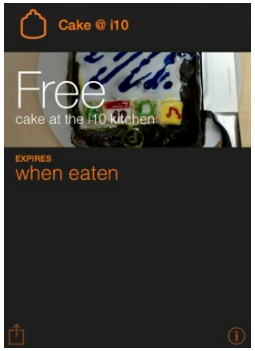

Social Framework

```
- (void) share
{
    NSString    *textToShare    = self.textField.text;
    UIImage     *imageToShare   = self.imageView.image;

    NSMutableArray *activityItems = [NSMutableArray array];
    if (textToShare)
        [activityItems addObject:textToShare];
    if (imageToShare)
        [activityItems addObject:imageToShare];

    UIActivityViewController *activityVC =
             [[UIActivityViewController alloc]
                  initWithActivityItems:activityItems
                  applicationActivities:nil];

    [self presentViewController:activityVC animated:YES completion:nil];
}
```



Demo

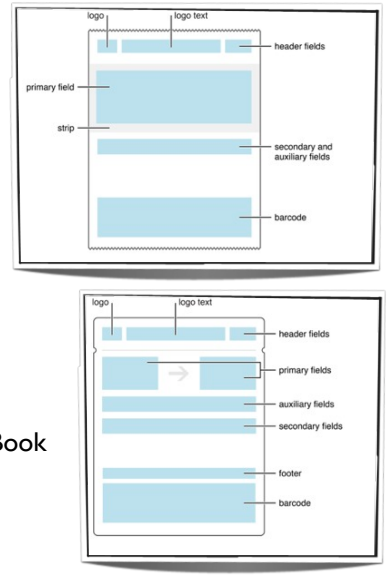
Passes For Users



- Installed from web or email resources
- Each pass represents a digital good
- Signed by Apple with the credentials of the vendor
- Pass shows human readable information
- Pass shows machine readable information
- PassKit presents passes in context

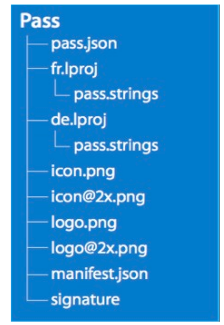
Passes Essentials

- Five predefined styles
  - Boarding pass
  - Coupon
  - Event Ticket
  - Store card
  - Generic
- Passes do not execute Code
- Are regularly checked for updates by PassBook App

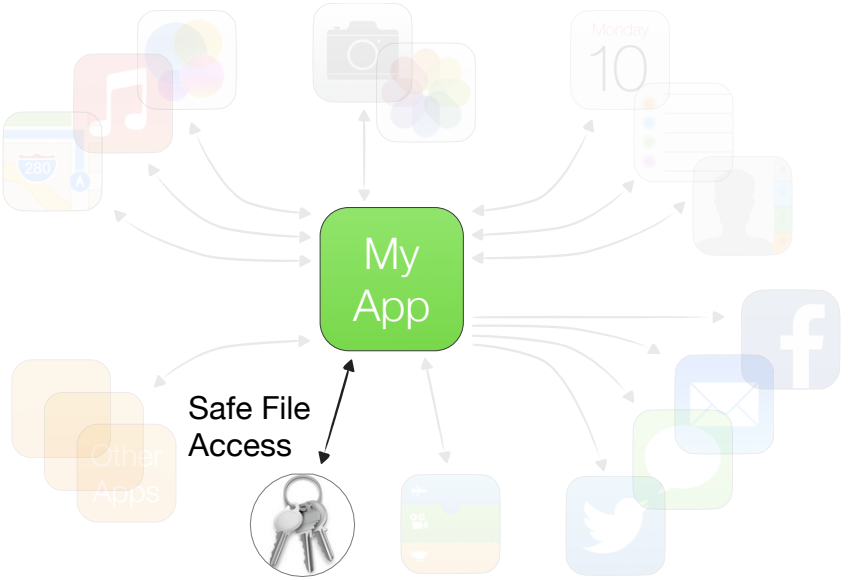


Passes Essentials

- Pass is a folder
  - JSON description of data and behavior
  - Image Resources
  - Translations
  - Signature
  - Zipped
- Behavior can include
  - Lifespan
  - Locations
  - Update URL



Demo: building a pass



Security issues you (your users) face

- Private information on devices
  - Is that a problem when the device gets out of hand?
  - Can the user tamper with it?
- Private communication over the air
  - Can an attacker gain information seeing the packets
  - Can an attacker fake a server?
- What cryptographic building blocks do we have available?
- What to do when we have no expertise?

Security Features that you can use

- Secure Communications
  - Https, SSL
  - Chain of Trust
- Secure File Storage
  - Data Protection
- Cryptographic Building Blocks
  - CommonCrypto
  - OpenSSL

Demo

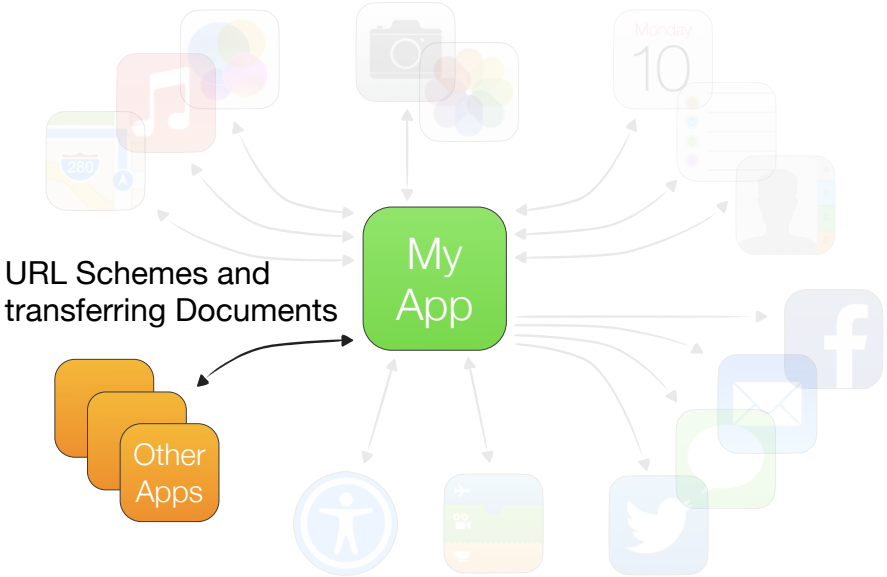
Security Features that you can use

- Secure File Storage
  - Data Protection

    When device is unlocked

    After the first unlock

    ...
- Cryptographic Building Blocks
  - CommonCrypto
  - OpenSSL



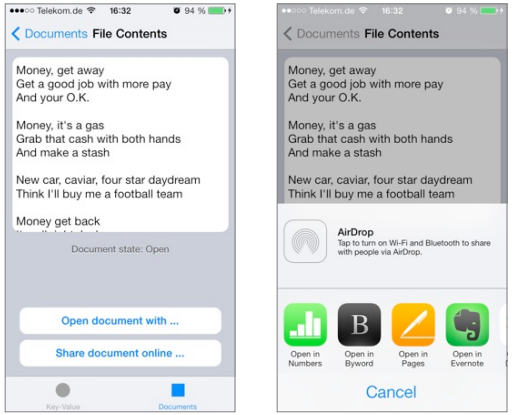
Option 1: Handing over documents

- How to get data from app to app?
  - "Open with..." ?
- UIDocumentInteractionController
  - Apps publish what files they can open
  - Sender app pushes document, user selects target app
  - Data is copied between app sandboxes
  - No way to track files

UIDocumentInteractionController

```
- (IBAction) openExternally:(id)sender
{
   // save first
   [self.document saveToURL:self.document.fileURL
          forSaveOperation:UIDocumentSaveForOverwriting
          completionHandler:^(BOOL success)
    {
        // bring up dialog from doc interaction controller
        self.docController = [UIDocumentInteractionController
                              interactionControllerWithURL:self.document.fileURL];

        BOOL didOpen = [docController presentOpenInMenuFromRect:CGRectZero
                                                          inView:self.openButton
                                                        animated:YES];
        if (!didOpen)
        {
            [[[UIAlertView alloc] initWithTitle:@"Cannot open file in other apps"
                                        message:@"Unfortunately, there is no app installed
                                                 that can handle this kind of file."
                                       delegate:nil
                              cancelButtonTitle:@"Ok"
                              otherButtonTitles:nil] show];
        }
    }];
}
```


UIDocumentController in Action

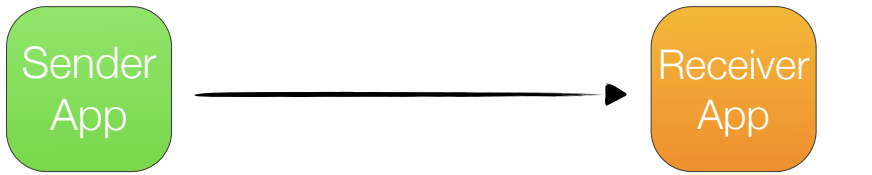


Check the iCloudPlayground demo code for more info

Option 2: Handing over URLs

```
// UIApplication
- (BOOL) openURL:(NSURL *)url
- (BOOL) canOpenURL:(NSURL *)url
```

```
// Application Delegate
- (BOOL) application:(UIApplication *)application
             openURL:(NSURL *)url
   sourceApplication:(NSString *)sourceApplication
          annotation:(id)annotation
```

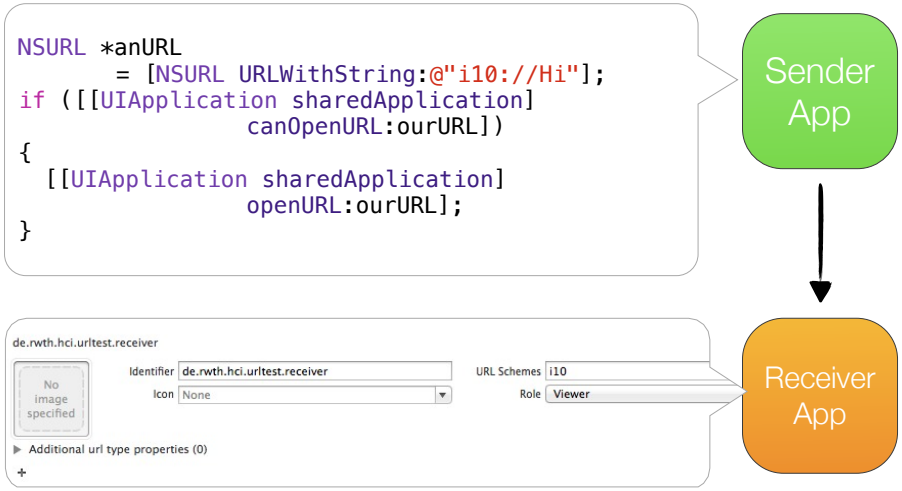


Built-in URLs

- `tel:024121050`
- `mailto:leonhard@lichtschlag.net`
- `mailto:leonhard@lichtschlag.net?cc=flo@cs.rwth-aachen.de&subject=I%20need%20more%20points`
- `http://maps.google.com/maps?daddr=Aachen,Germany`
- `http://www.youtube.com/watch?v=QH2-TGUlwu4`
- `prefs:root=General&path=Bluetooth`

Receiving Custom URLs

```
NSURL *anURL
       = [NSURL URLWithString:@"i10://Hi"];
if ([[UIApplication sharedApplication]
                  canOpenURL:ourURL])
{
  [[UIApplication sharedApplication]
                  openURL:ourURL];
}
```



Demo

Summary

- Further Reading:
  - EventKit Programming Guide
  - Address Book Programming Guide
  - Calendar and Reminders Programming Guide
  - Camera Programming Topics for iOS
  - System Messaging Topics for iOS
  - PassKit Programming Guide
  - Document Interaction Programming Topics for iOS