Advanced smartphone forensics

Apple iCloud: backups, document storage, keychain

BlackBerry 10 backup encryption

Vladimir Katalov, ElcomSoft Co. Ltd

Smartphone forensics methods

	iOS	WP8	BB 10
Logical acquisition	Yes(*)	No	No
Advanced logical	Yes	No	No
Physical acquisition	Yes/No(**)	No	No
Chip-off	No	?	?
Local backup	Yes	No	Yes
Cloud backup	Yes	Yes/No(***)	No
iCloud keychain	Yes	N/A	N/A
Documents in cloud	Yes	Yes	No
Location service	Yes	Yes	No

(*) In fact, same as local backup
(**) For A5+ devices, subject of jailbreak availability
(***) No complete backup, only selected categories

iOS forensics

- **Logical acquisition**
  - "Ask" device to produce backup
  - Device must be unlocked(*)
  - Device may produce encrypted backup
  - Limited amount of information (but more than you think)
- **Physical acqusition**
  - Boot-time exploit to run unsigned code or jailbreak
  - Device lock state isnt relevant, can bruteforce passcode
  - Can get all information from the device (incl. deleted data)
- **Advanced logical acquisition**
  - By direct access to some services running on the iPhone
  - Device must be unlocked*
  - Limited amount of information (some as in local backup, but plus something extra)
  - Backup password isn't relevant
  - Can be performed over Wi-Fi
- **iCloud**
  - Need Apple ID and password (or *authentication token*)
  - Can be performed without the device itself
  - Almost the same information as in local backup
  - Can get the documents and location data, too

** But there is a workaround ;)*

Backups – what and when

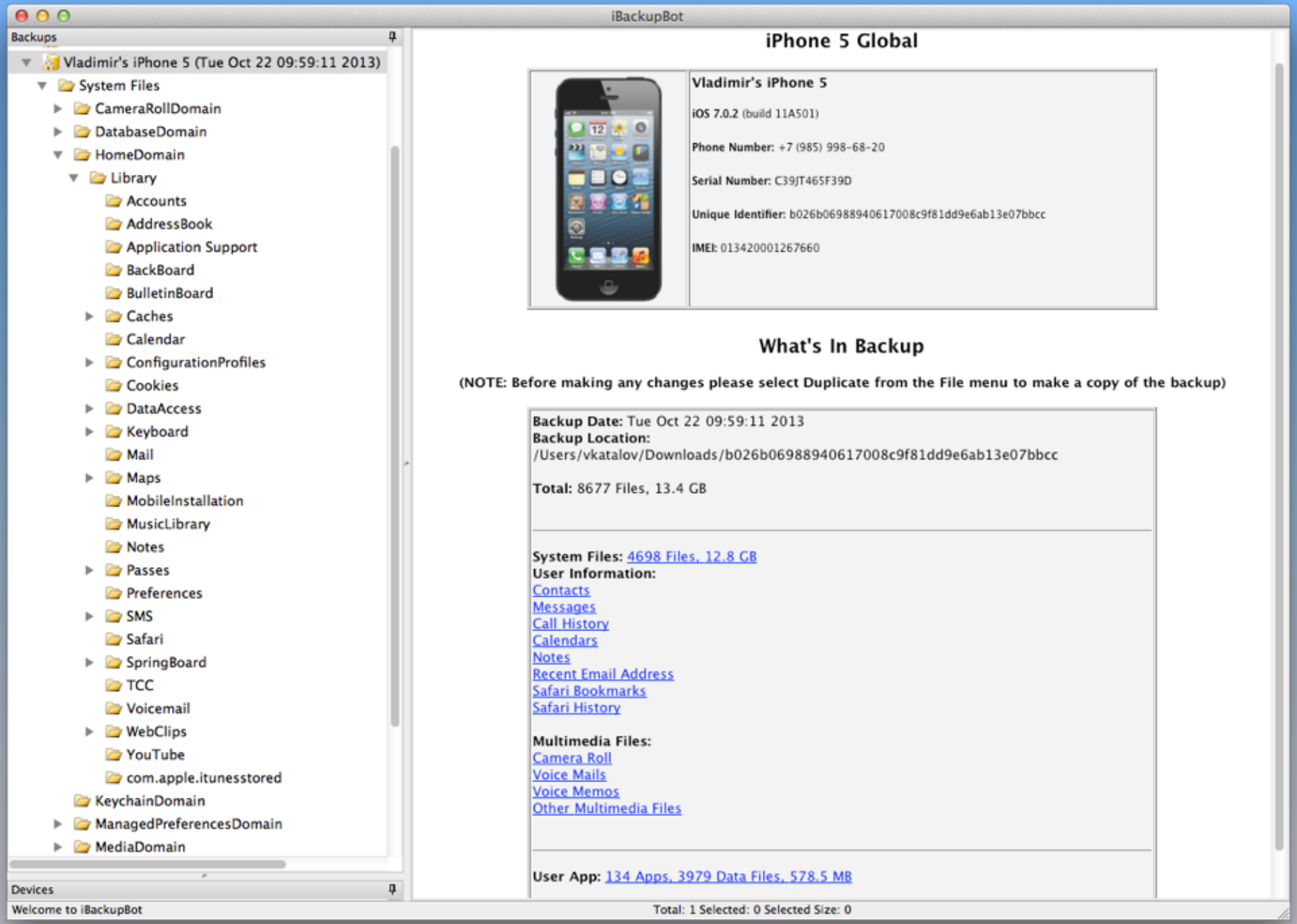
- *Contacts and Contact Favorites*
- *Messages (including iMessages)*
- *Call history*
- *Application data*
- *Device settings*
- *Camera roll (photos and videos)*
- *Purchases (music, movies, TV, apps, books)*
- *Mail accounts*
- *Network settings (saved Wi-Fi hotspots, VPN settings etc)*
- *Paired Bluetooth devices*
- *Safari bookmarks, cookies, history*
- *... and much more*

★ **Local backups**

- iTunes create backups when:
  - Sync with iTunes
  - [File] | [Devices backup]

★ **iCloud backups**

- Backup runs daily when device is:
  - Connected to the Internet over Wi-Fi
  - Connected to a power source
  - Locked
- Can force backup
  - [Settings] | [iCloud] | [Storage & Backup] | [Back Up Now]


iBackupBot
Backups
Vladimir's iPhone 5 (Tue Oct 22 09:59:11 2013)
System Files
CameraRollDomain
DatabaseDomain
HomeDomain
Library
Accounts
AddressBook
Application Support
BackBoard
BulletinBoard
Caches
Calendar
ConfigurationProfiles
Cookies
DataAccess
Keyboard
Mail
Maps
MobileInstallation
MusicLibrary
Notes
Passes
Preferences
SMS
Safari
SpringBoard
TCC
Voicemail
WebClips
YouTube
com.apple.itunesstored
KeychainDomain
ManagedPreferencesDomain
MediaDomain
Devices
Welcome to iBackupBot
iPhone 5 Global
Vladimir's iPhone 5
iOS 7.0.2 (build 11A501)
Phone Number: +7 (985) 998-68-20
Serial Number: C39JT465F39D
Unique Identifier: b026b06988940617008c9f81dd9e6ab13e07bbcc
IMEI: 013420001267660
What's In Backup
(NOTE: Before making any changes please select Duplicate from the File menu to make a copy of the backup)
Backup Date: Tue Oct 22 09:59:11 2013
Backup Location:
/Users/vkatalov/Downloads/b026b06988940617008c9f81dd9e6ab13e07bbcc
Total: 8677 Files, 13.4 GB
System Files: 4698 Files, 12.8 GB
User Information:
Contacts
Messages
Call History
Calendars
Notes
Recent Email Address
Safari Bookmarks
Safari History
Multimedia Files:
Camera Roll
Voice Mails
Voice Memos
Other Multimedia Files
User App: 134 Apps, 3979 Data Files, 578.5 MB
Total: 1 Selected: 0 Selected Size: 0

Wait, there is more…

- **Google Apps data: Search, Maps, YouTube etc**

AppDomain-com.google.*

- **Social networking & communications**

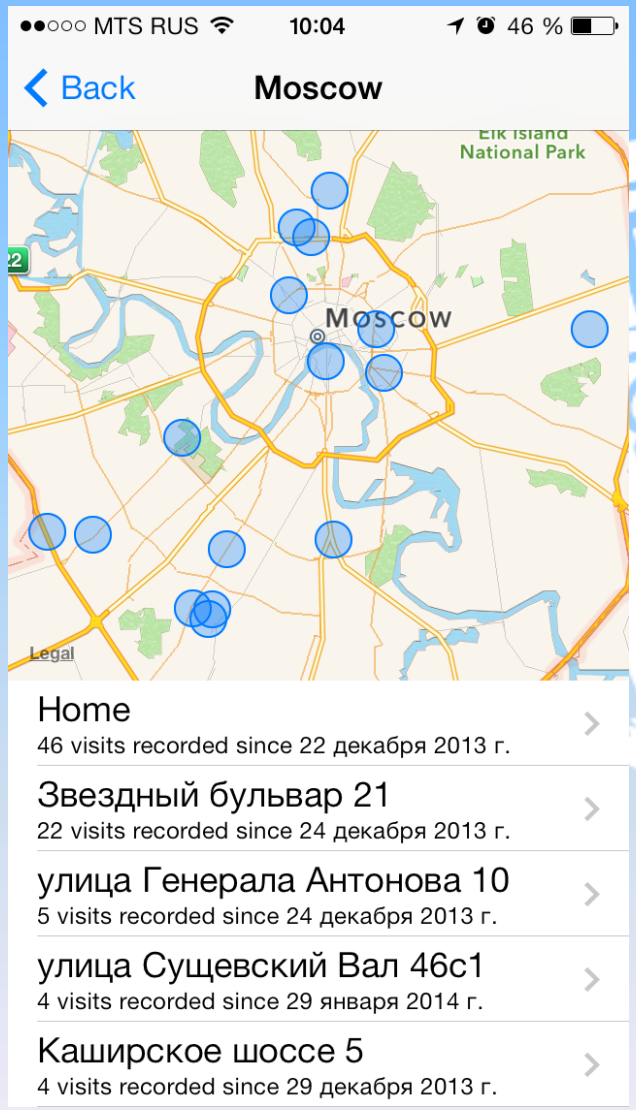
AppDomain-net.whatsapp.WhatsApp*
AppDomain-com.burbn.instagram*
AppDomain-com.facebook.Facebook*
AppDomain-com.facebook.Messenger*
AppDomain-com.skype.skype*
AppDomain-com.atebits.Tweetie2*
AppDomain-com.linkedin.LinkedIn*
AppDomain-com.naveenium.foursquare*
AppDomain-com.viber*

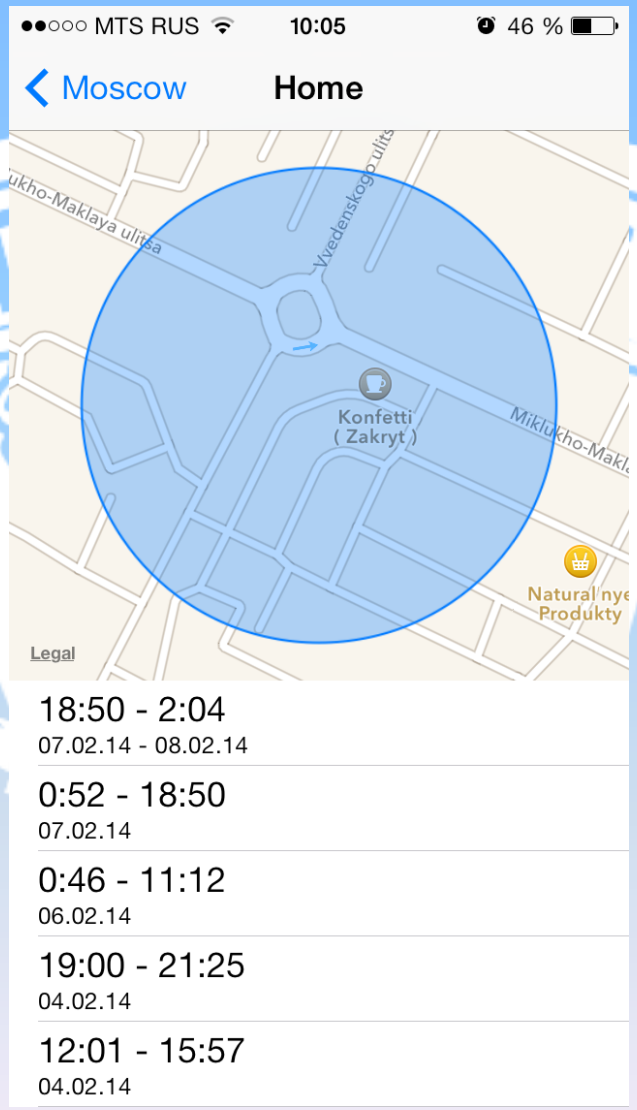
- **Other**

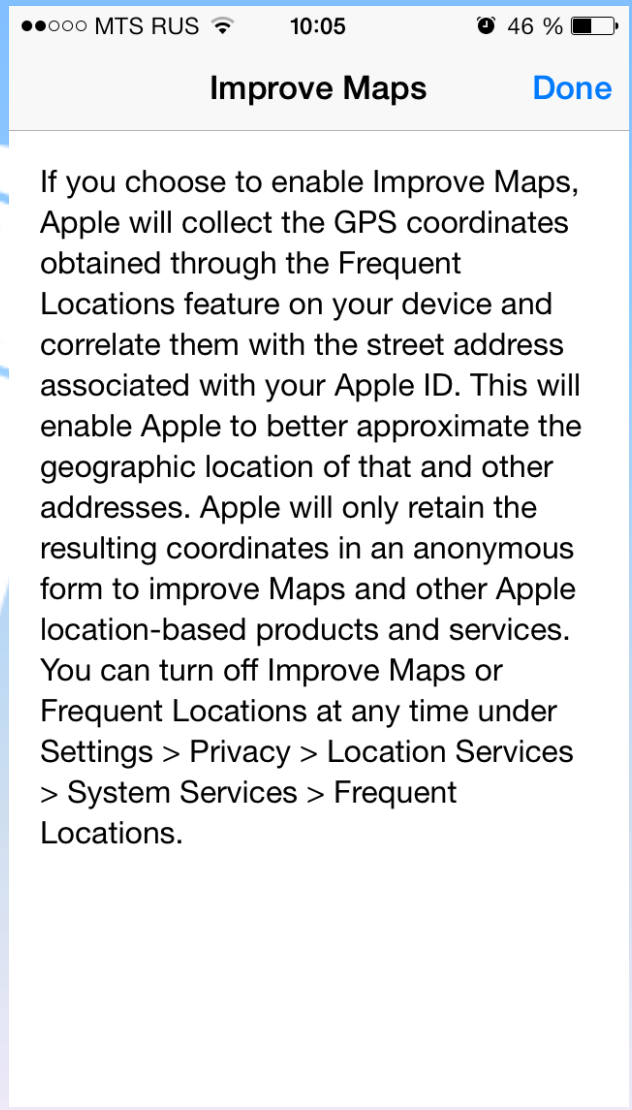
HomeDomain\Library\Keyboard*
HomeDomain\Library\Passes*
HomeDomain\Library\Voicemail*
HomeDomain\Library\Maps*

- Message attachments (even from deleted conversations!)
- Pictures from twitter posts
- Last backup date & time
- Info on Wi-Fi access points you ever connected to (SSID, security, signal etc)
- ... a lot of other interesting stuff :)

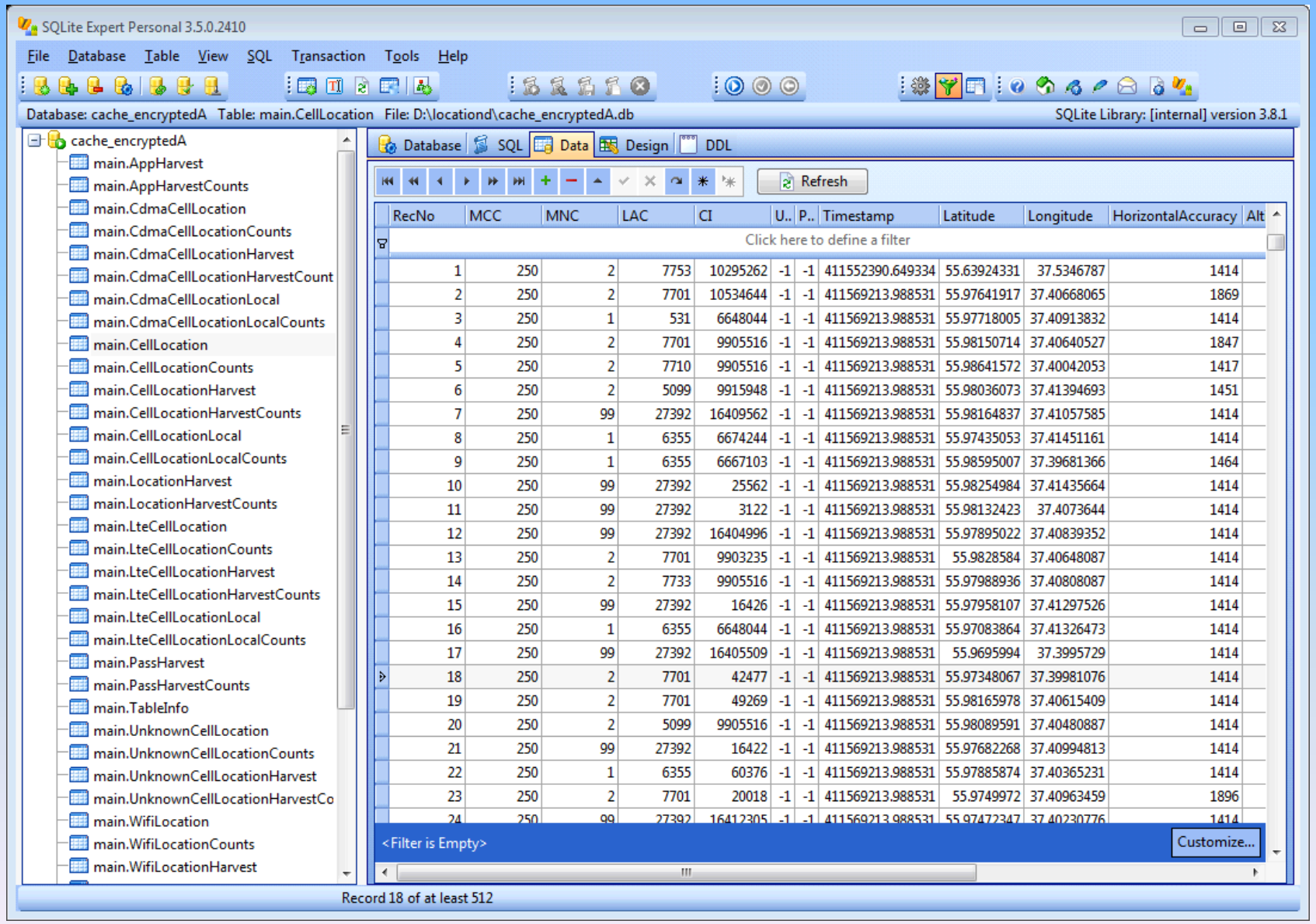
Frequent locations



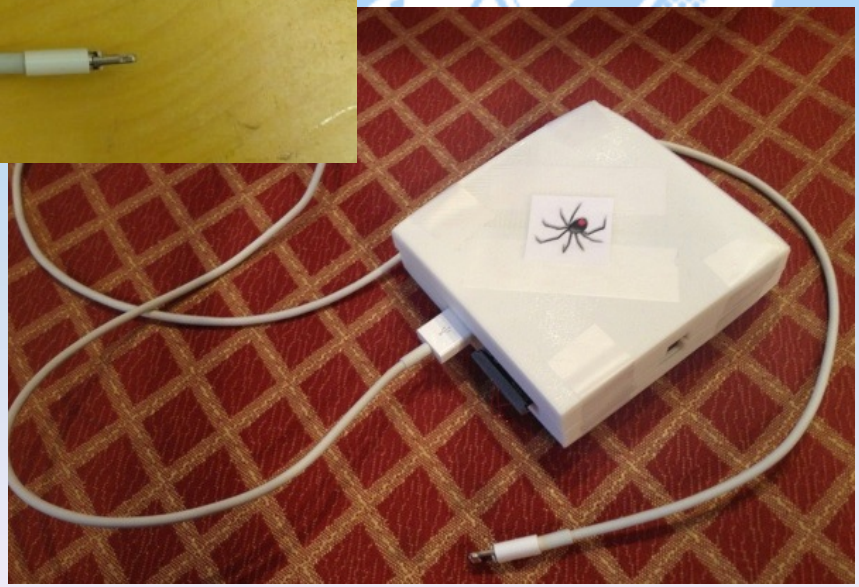


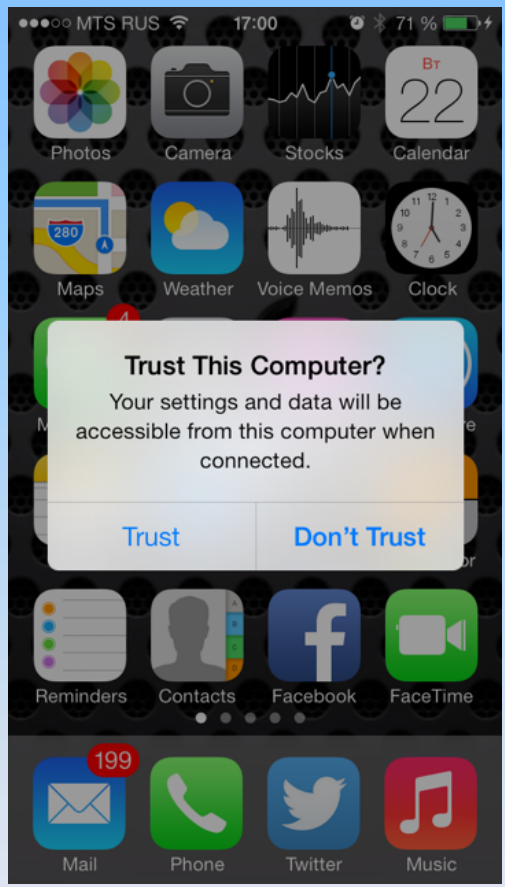


Location services



Do you really trust your charger?



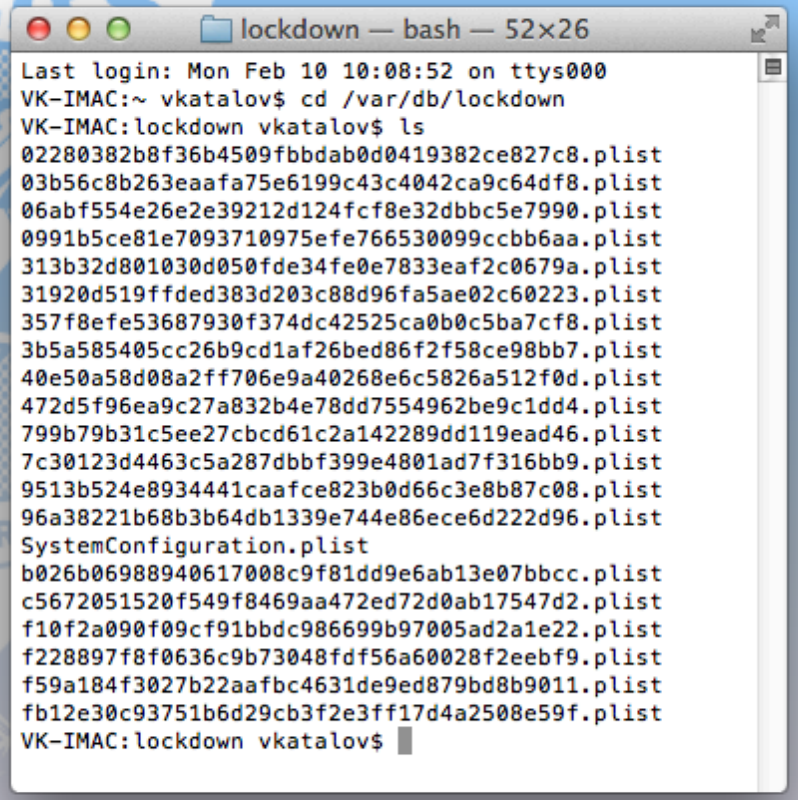


Pair-locking

- iOS device: /var/root/Library/Lockdown
- Mac: /var/db/lockdown

lockdownd service

- backup service
- software installation service
- get device name & UDID
- sync data
- retrieve a screenshot
- request iOS diagnostic information
- put device into recovery mode
- manage provisioning profiles



Advanced logical acqusition

- *for jailbroken devices - the entire file system*
- device information: name, model, IMEI, UUID, serial number etc
- all the media (photos, videos, iTunes library, iBooks)
- application data (including temporary files and *caches* folder)
- various device settings
- log files and diagnostic information
- cached web data (e.g. pictures from social networks)
- keyboard typing caches
- SMS and iMessages (including attachments, even to deleted messages)
- address book
- calendar
- voice mail
- WAL (Write-Ahead Logging) files for most SQLite databases
- ...and more

Works even if device is passcode-locked and backup encryption is set

Can be done over Wi-Fi

Only need the pairing record

iCloud Control Panel

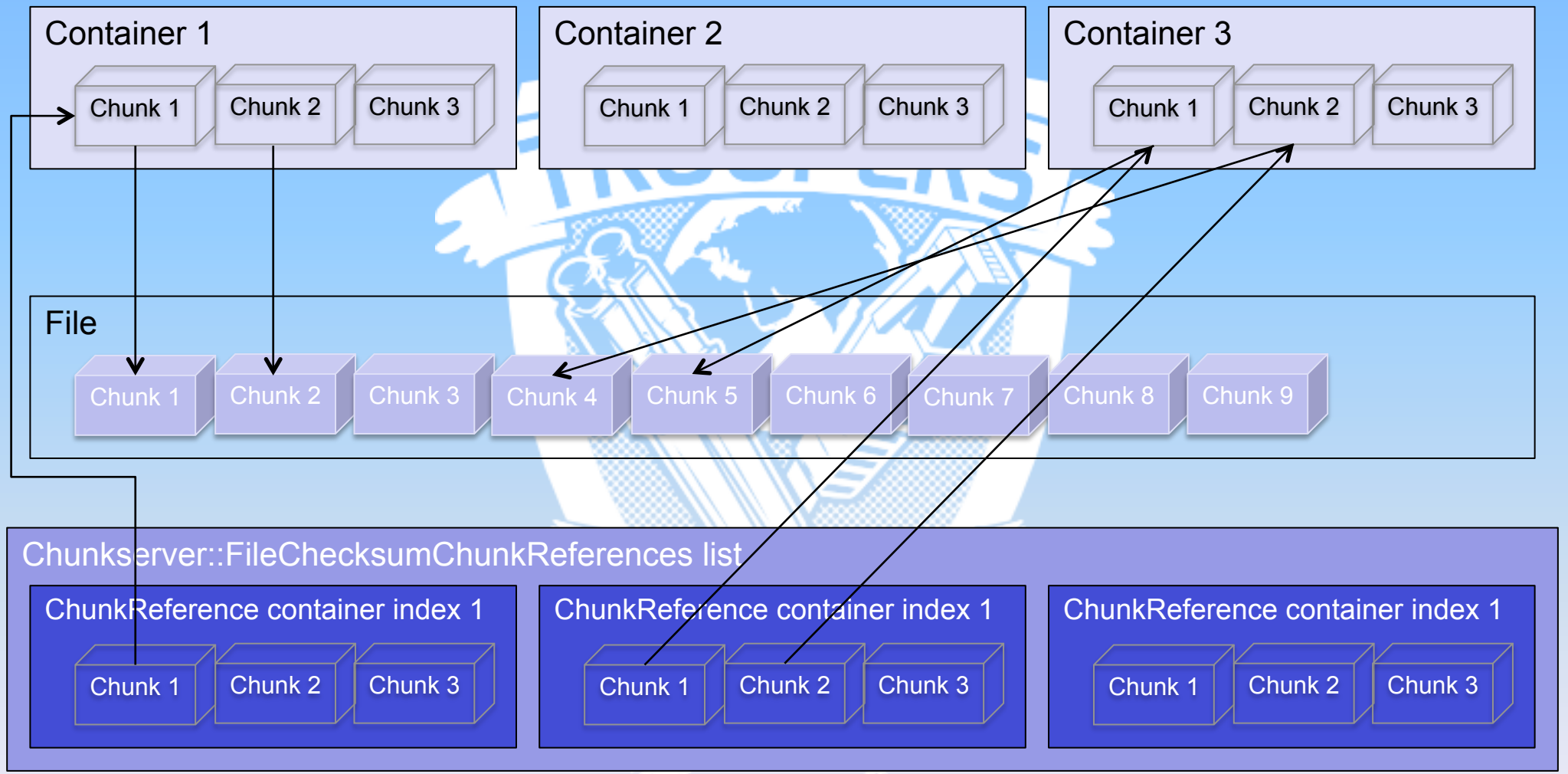
iCloud backups reverse engineering

- no backup to iCloud from iTunes :(

so...

- jailbreak iPhone
- Install Open SSH, get keychain (keychain-2.db)
- [Settings] | [iCloud] | [Delete Account] | [Delete from My iPhone]
- [Settings] | [General] | [Reset] | [Reset All Settings]
- reboot
- set up Wi-Fi connection (proxy)
- replace keychain with our own trusted root certificate
- ... read all the traffic :)

Files in iCloud



iCloud backups – authentication, get token, get keys

Authentication:
https://setup.icloud.com/setup/authenticate/$APPLE_ID$
Authorization:Basic <authentication data>
authentication data = mime64 (AppleID:password)
returns: mmeAuthToken, dsPrsID

https://setup.icloud.com/setup/get_account_settings
Authorization:Basic <authentication data>
authentication data = mime64 (dsPrsID:mmeAuthToken)
returns: mmeAuthToken (new/other one!!)

https://p11-mobilebackup.icloud.com/mbs/(dsPrsID)
Authorization: <authentication data>
authentication data = mime64 (dsPrsID:mmeAuthToken)
returns: list of backup IDs (backupudid)

https://p11-mobilebackup.icloud.com/mbs/7052-4754-8032-4429-3092/(backupudid)/getKeys

Get snaphots, file auth. tokens, chunks URLs

Enumerate snapshots

HTTPS GET
https://p11-mobilebackup.icloud.com/mbs/(dsPrsID) /(backupudid)/(snapshotid)/listFiles?[…]

Get file authentication tokens

HTTPS POST
https://p11-mobilebackup.icloud.com/mbs/(dsPrsID)/(backupudid)/(snapshotid)/getFiles

Get URLs for file chunks

HTTPS POST
https://p11-content.icloud.com/(dsPrsID)/authorizeGet

Download chunks

Windows Azure:
http://msbnx000004.blob.core.windows.net:80/cnt/g6YMJKQBPxQruxQAr30C?[...]
Amazon AWS:
http://us-std-00001.s3-external-1.amazonaws.com/l9rh20QBPX4jizMAr3vY?[...]

iCloud backups – data encryption

- get keyData from iCloud
- wrappedOffset = keyDataSize - (ECP_LEN + WRAPPED_KEY_LEN)
- get wrappedKey (at wrappedOffset)
- get CLASS_KEY
- iOS 5/6: ((UINT32*)(keyData + wrappedOffset))[-1]
- iOS 7: ((UINT32*)(keyData + wrappedOffset))[-3]
- ***iOS 7.1: key type (2/3/4), two keys***
- decrypt wrappedKey using CLASS_KEY
- get AES_KEY from wrappedKey
- file decryption: by 0x1000 blocks (unique IV for every block)

```
#define HFS_IV_GENERATOR 0x80000061
#define IV_GEN(x) (((x) >> 1) ^ (((x) & 1) ?
HFS_IV_GENERATOR : 0))

static  UInt8 *genIV (UInt32 seed, void *pIV) {
  UInt32 *pdw = (UInt32*)pIV;
  pdw[0] = seed = IV_GEN(seed);
  pdw[1] = seed = IV_GEN(seed);
  pdw[2] = seed = IV_GEN(seed);
  pdw[3] = seed = IV_GEN(seed);
  return (UInt8*)pIV;
}
```

```
AES_KEY aesIV;
makeIVkey (&aesIV, abKey, SYSTEM_KEY_LEN);
(abKey is AES_KEY we have got from wrappedKey)

static AES_KEY *makeIVkey (AES_KEY *pAES, UInt8 *pb,
size_t cb) {
   SHA_CTX sha;
   UInt8 abHash[SHA_DIGEST_LENGTH];
   SHA1_Init (&sha);
   SHA1_Update (&sha, pb, cb);
   SHA1_Final (abHash, &sha);
   AES_set_encrypt_key (abHash, 128, pAES);
   return pAES;
}
```

iCloud backups – access without Apple ID & password

Windows:
%user%\AppData\Roaming\Apple Computer\Preferences\ByHost\com.apple.AOSKit.{guid}.plist

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>qtt.test@icloud.com</key>
  <dict>
    <key>data</key>
    <data>
    AQAAANCMnd8BFdER … 0VbqsqAdjn+o+w==
    </data>
    <key>version</key>
    <string>1</string>
  </dict>
</dict>
</plist>
```

Mac:
/Library/Application Support/iCloud/Accounts/(dsid)

- Get GenericPassword from local keychain (by AppleID)
- Decode GenericPassword (base64)
- Make hmac-md5 digest using key:
  "\x74\x39\x73\x22\x6C\x78\x5E\x61\x77\x65\x2E\x35\x38\x30\x47\x6A\
  \x25\x27\x6C\x64\x2B\x30\x4C\x47\x3C\x23\x39\x78\x61\x3F\x3E\x76\
  \x62\x29\x2D\x66\x6B\x77\x62\x39\x32\x5B\x7D\x00"
- Decrypt file (AES-128 in PKCS7Padding mode)

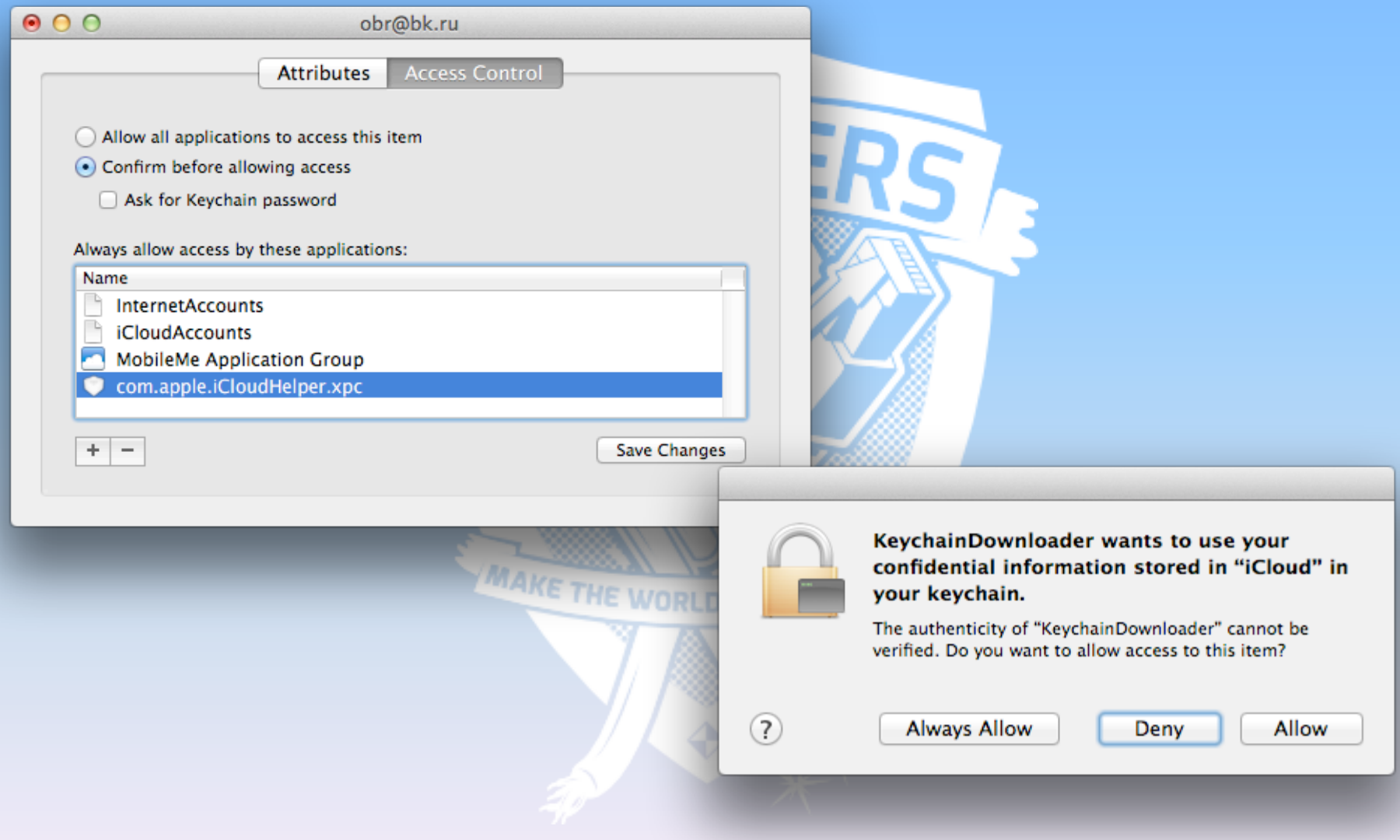
iCloud backups: decrypt auth token

```
BOOL __stdcall CryptProtectData(
    DATA_BLOB *pDataIn,
    LPCWSTR szDataDescr, // 0
    DATA_BLOB *pOptionalEntropy, // hardcoded in AOSKit.dll
    PVOID pvReserved, // 0
    CRYPTPROTECT_PROMPTSTRUCT *pPromptStruct, // 0
    DWORD dwFlags, // 0
    DATA_BLOB *pDataOut); // result
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>appleAccountInfo</key>
  <dict>
    <key>appleId</key>
    <string>qtt.test@icloud.com</string>
    <key>dsPrsID</key>
    <string>1695808496</string>
    <key>firstName</key>
    <string>Vladimir</string>
    <key>lastName</key>
    <string>Katalov</string>
    <key>primaryEmail</key>
    <string>qtt.test@icloud.com</string>
  </dict>
<key>tokens</key>
<dict>
  <key>mmeAuthToken</key>
  <string>AQAAAABS9K/N5WTDdAlOHT5G9vptut2yJsUS4el=</string>
</dict>
</plist>
```

```
BYTE* entropy = (BYTE*)"\
\x1D\xAC\xA8\xF8\xD3\xB8\x48\x3E\x48\x7D\x3E\x0A\x62\x07\xDD\x26\
\xE6\x67\x81\x03\xE7\xB2\x13\xA5\xB0\x79\xEE\x4F\x0F\x41\x15\xED\
\x7B\x14\x8C\xE5\x4B\x46\x0D\xC1\x8E\xFE\xD6\xE7\x27\x75\x06\x8B\
\x49\x00\xDC\x0F\x30\xA0\x9E\xFD\x09\x85\xF1\xC8\xAA\x75\xC1\x08\
\x05\x79\x01\xE2\x97\xD8\xAF\x80\x38\x60\x0B\x71\x0E\x68\x53\x77\
\x2F\x0F\x61\xF6\x1D\x8E\x8F\x5C\xB2\x3D\x21\x74\x40\x4B\xB5\x06\
\x6E\xAB\x7A\xBD\x8B\xA9\x7E\x32\x8F\x6E\x06\x24\xD9\x29\xA4\xA5\
\xBE\x26\x23\xFD\xEE\xF1\x4C\x0F\x74\x5E\x58\xFB\x91\x74\xEF\x91";
```

iCloud backups: decrypt auth token (cont-d)



iCloud backups – encryption, summary

iCloud encryption

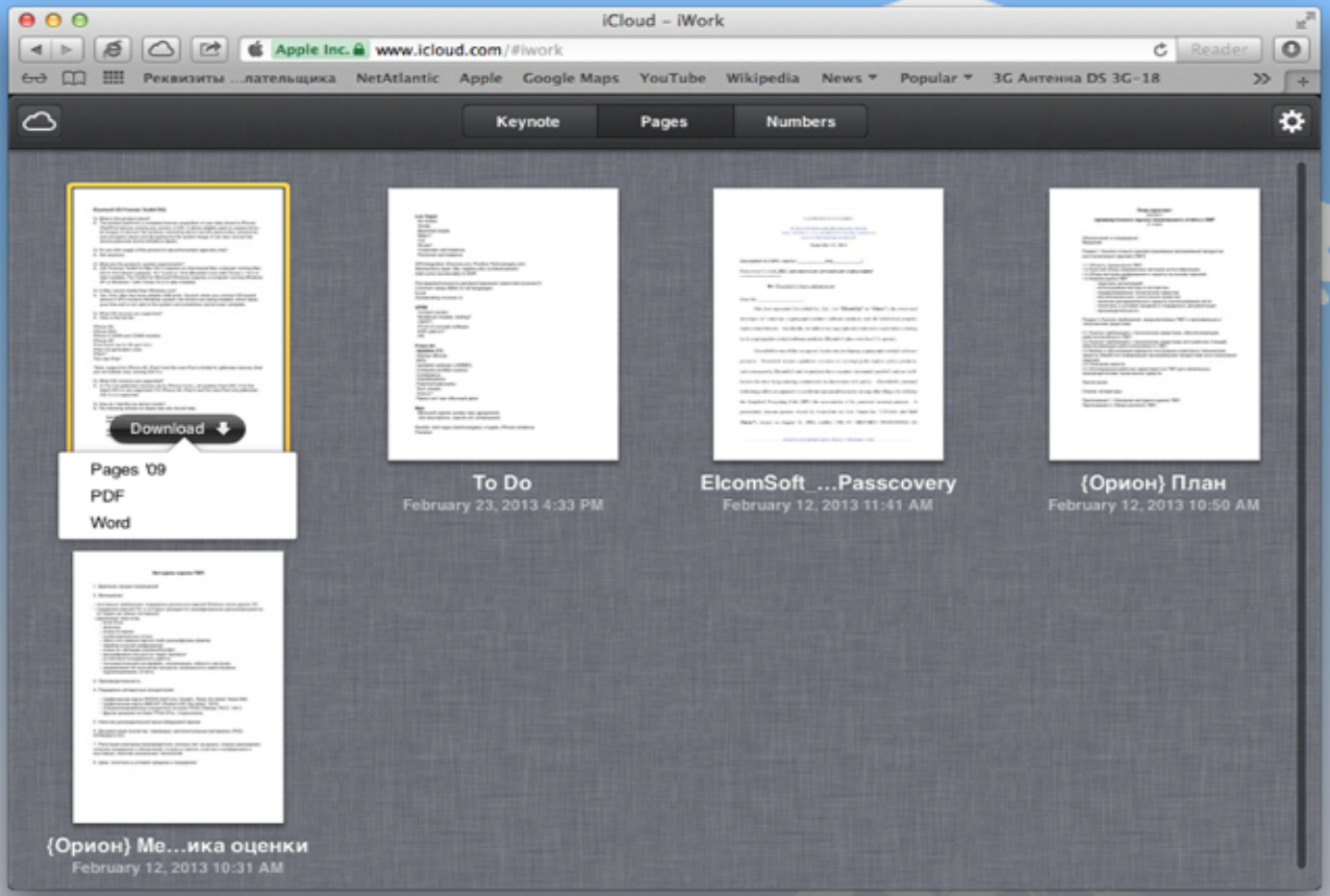
- Data stored at 3rd-party storage providers is encrypted
- Apple has encryption keys to that data
- Few files are further encrypted using keys from OTA backup keybag
- Keychain items are encrypted using keys from OTA backup keybag
- Need key 0x835 (securityd) to decrypt most keys from OTA backup keybag

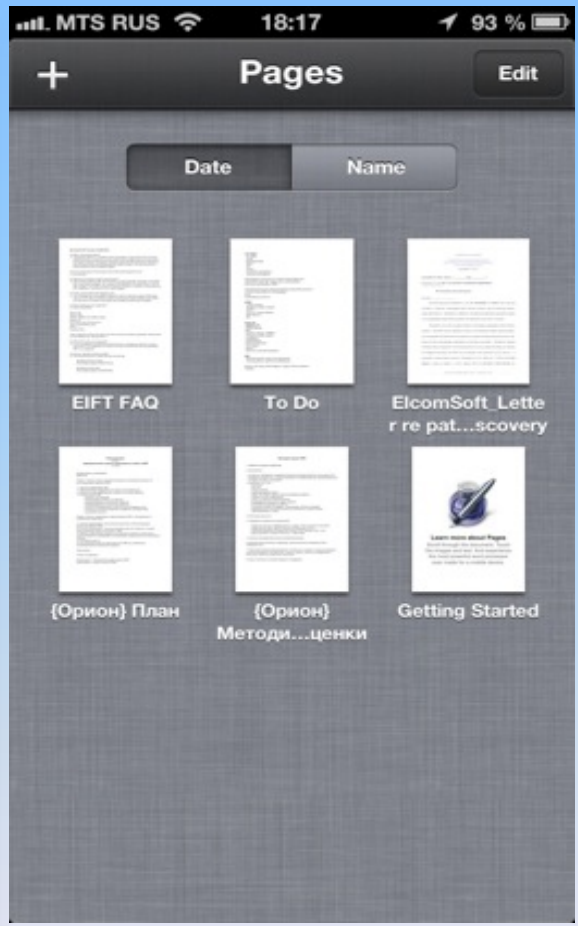
iCloud backups – summary

- There is no user-configurable encryption for iCloud backups
- iCloud backups are stored in Microsoft and Amazon clouds in encrypted form
- Apple holds encryption keys and thus have access to data in iCloud backups
- If Apple stores 0x835 keys then it can also have access to Keychain data
- Apple may have legal obligations to do this (e.g. LE)

- *No notification after backup downloading (as with device restore)!*

iCloud documents





Get files from iCloud

To get list of files

- Authentication request (with given AppleID & password). Client gets mmeAuthToken in return; which, in order, is used to create authentication token (together with dsid). dsid (Destination Signaling IDentifier) is an unique ID assigned to the user when registering at iCloud.com.

- Request to get AccountSettings. Client gets an URL (ubiquityUrl) with an address to get UUID (unique user identifier), file list, info on file tokens and for authorization.

- Request to get file list (POST). Output (for every file):
  - file name
  - file id
  - parent folder id
  - last change time
  - checksum
  - access rights

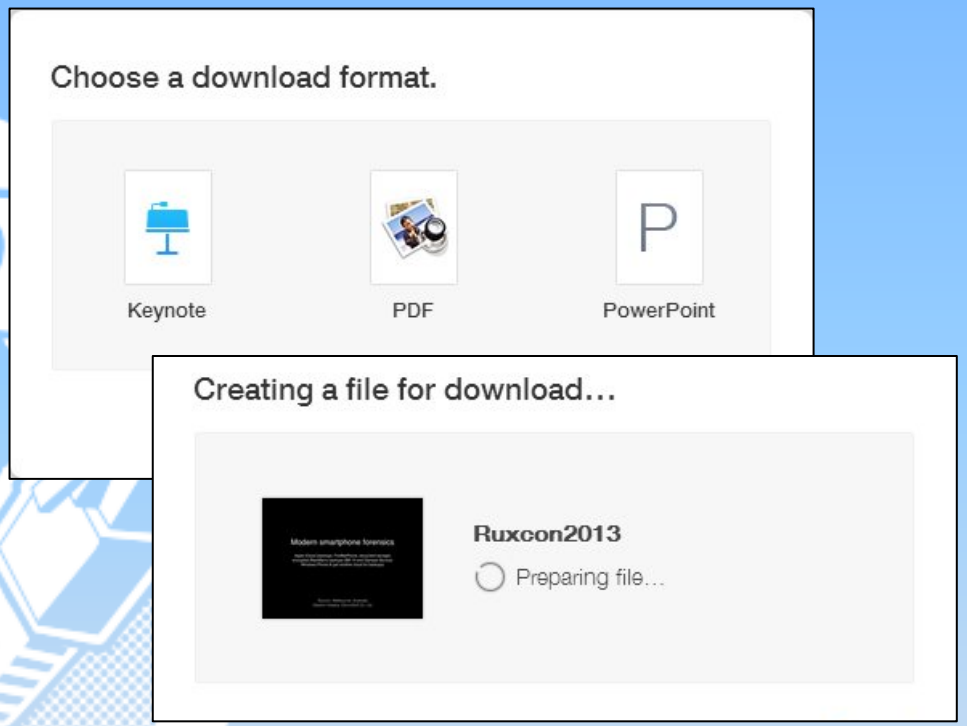
To download given file

- Request to get file token (using file id, checksum and aliasMap).

- Authorization request. Returns information on file chunks and containers. Output: container list (with URLs) and chunk information.

iCloud files: packages

- KeyNote: PDF, Microsoft PowerPoint, KeyNote
- Pages: PDF, Microsoft Word, Pages
- Numbers: PDF, Microsoft Excel, Numbers
- Some other programs: 1Password, WhatsApp etc



Storage: plist + content (text, media files)

Validate, login
https://setup.icloud.com/setup/ws/1/validate
https://setup.icloud.com/setup/ws/1/login

Export
https://p15-ubiquityws.icloud.com/iw/export/(dsid)/export_document?...

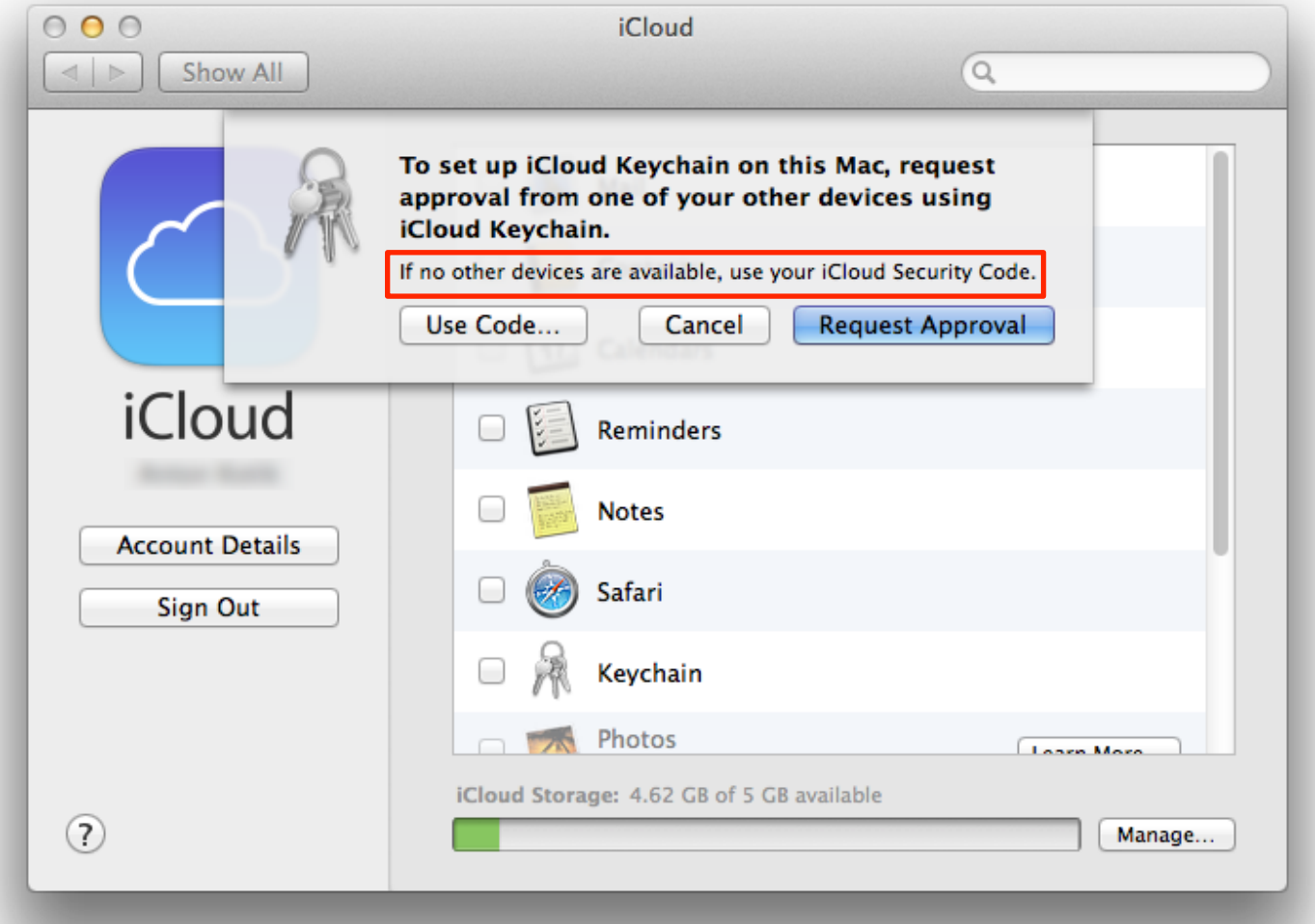
Check export status
https://p15-ubiquityws.icloud.com/iw/export/(dsid)/check_export_status?...

Download converted file
https://p15-ubiquityws.icloud.com/iw/export/(dsid)/download_exported_document?

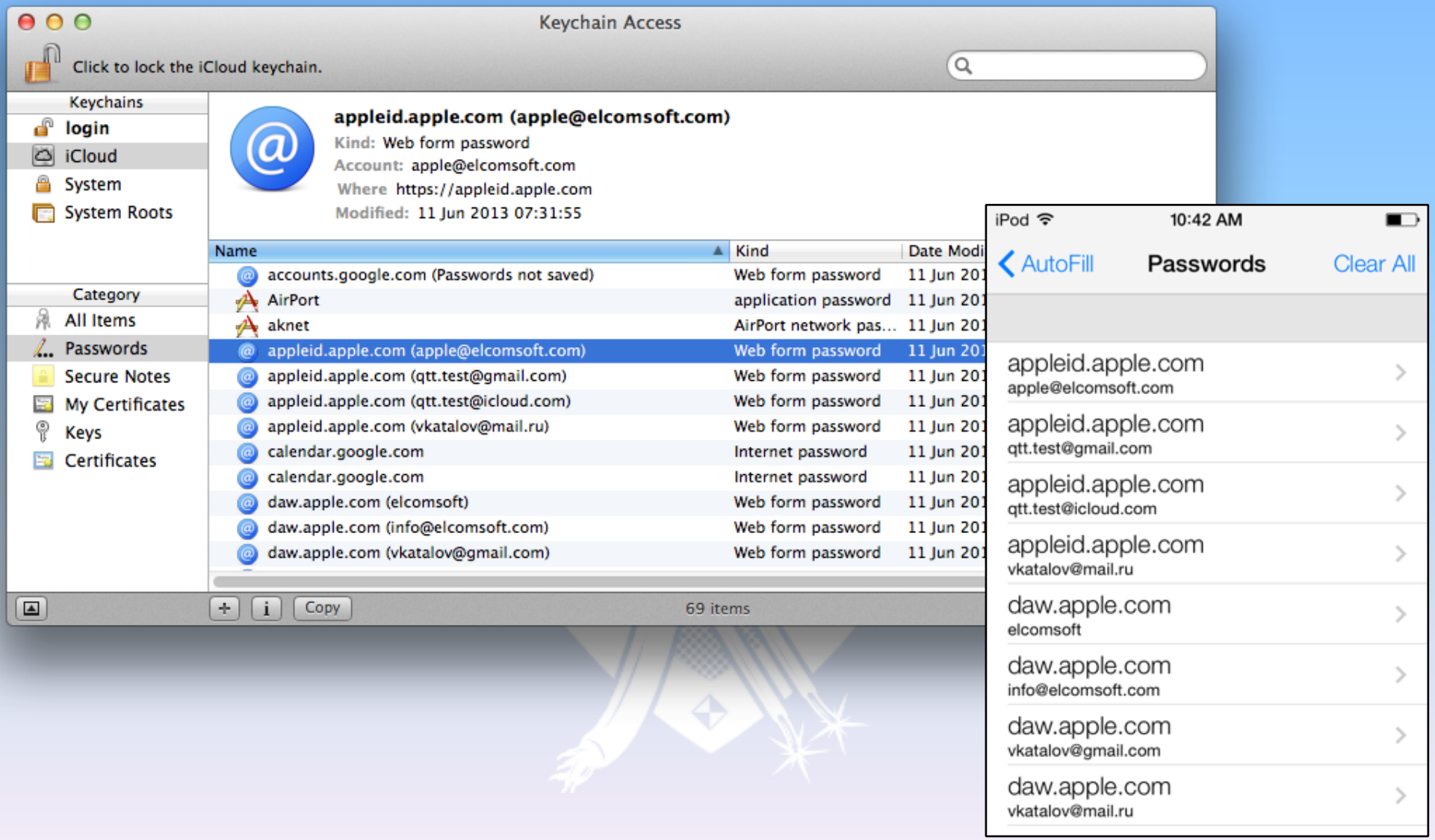
iWork'2013 files w/password: only available as packages
.iwph: password hint, .iwph2: password hash (pbkdf2(sha1, 100000))

iCloud keychain

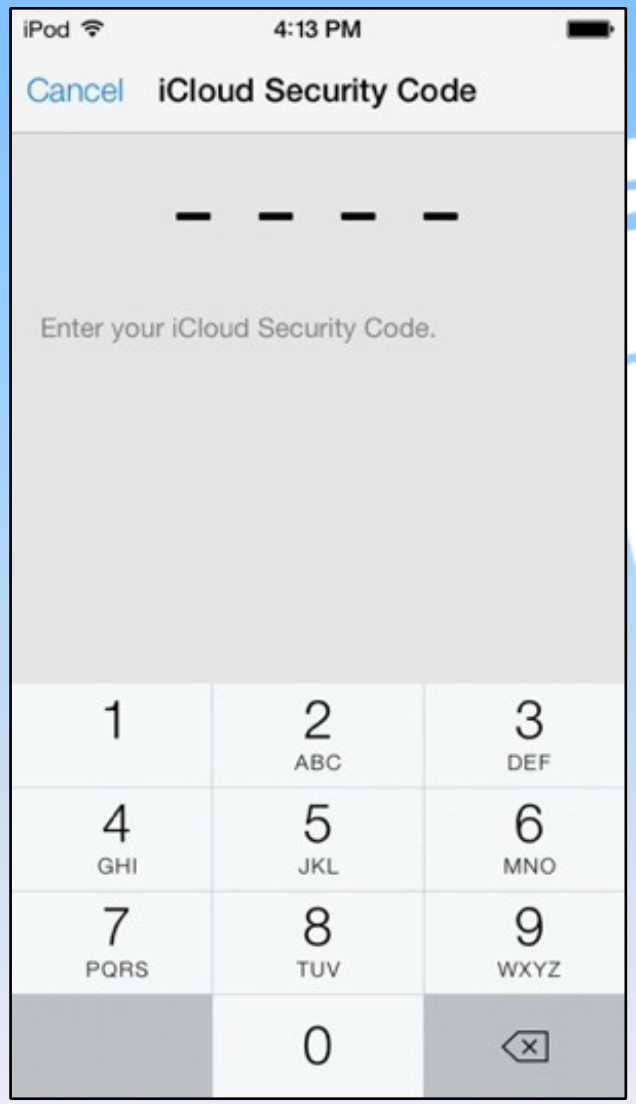
iCloud keychain

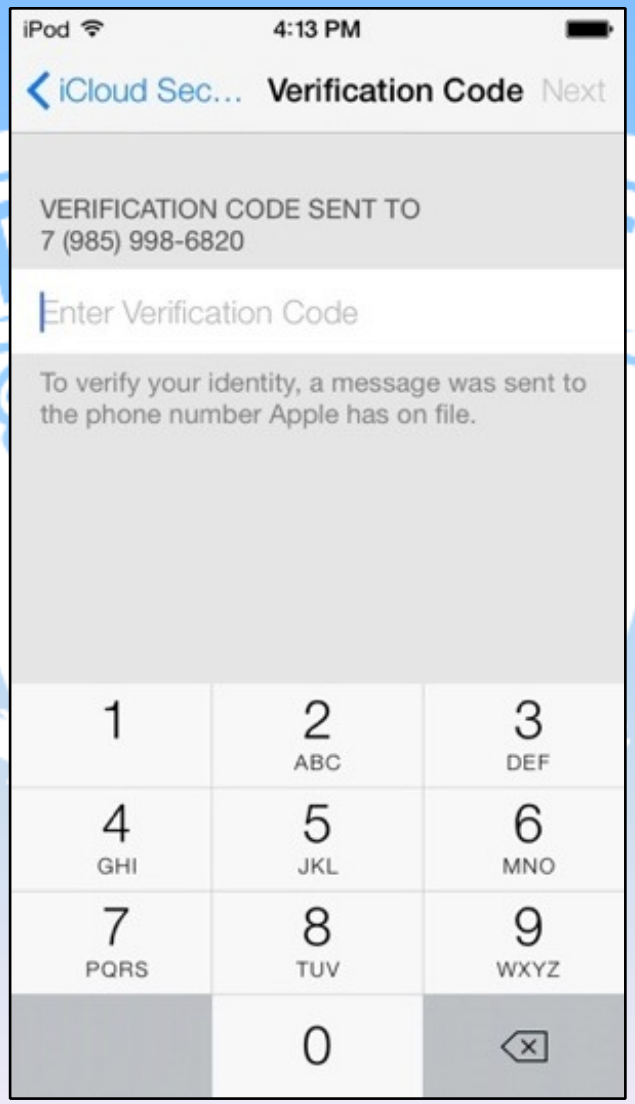


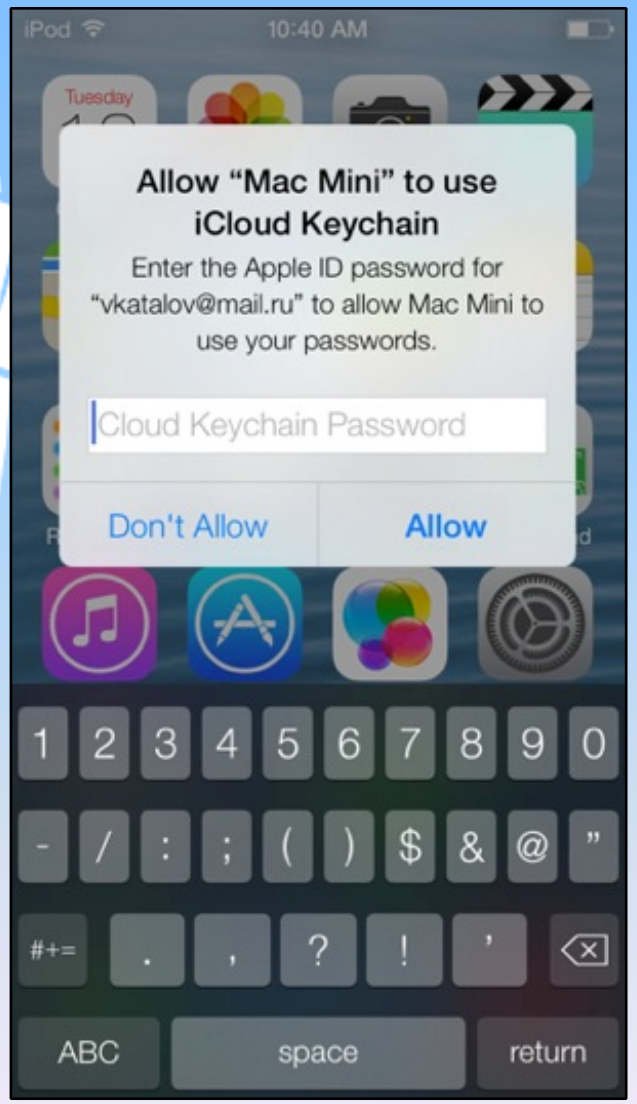
iCloud keychain (cont'd)



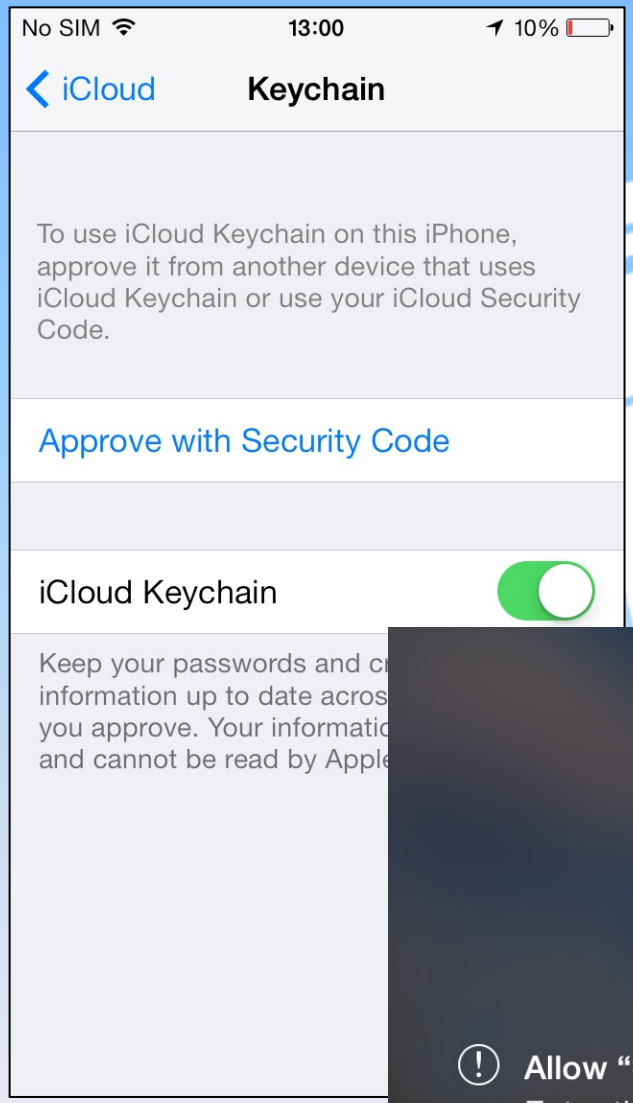
iCloud keychain (cont'd)

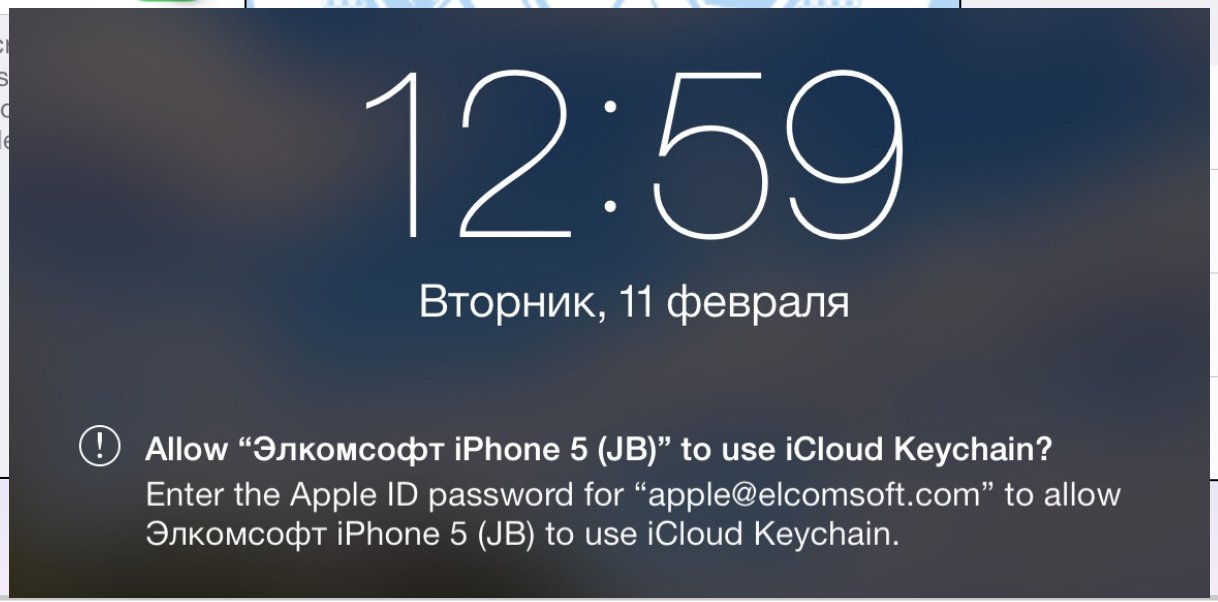


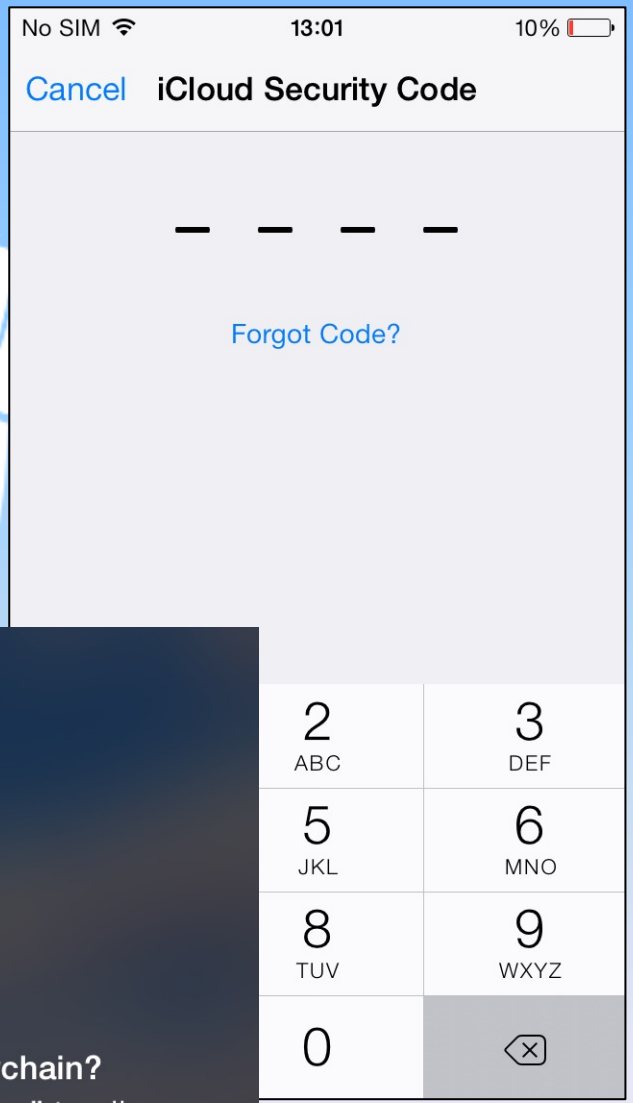




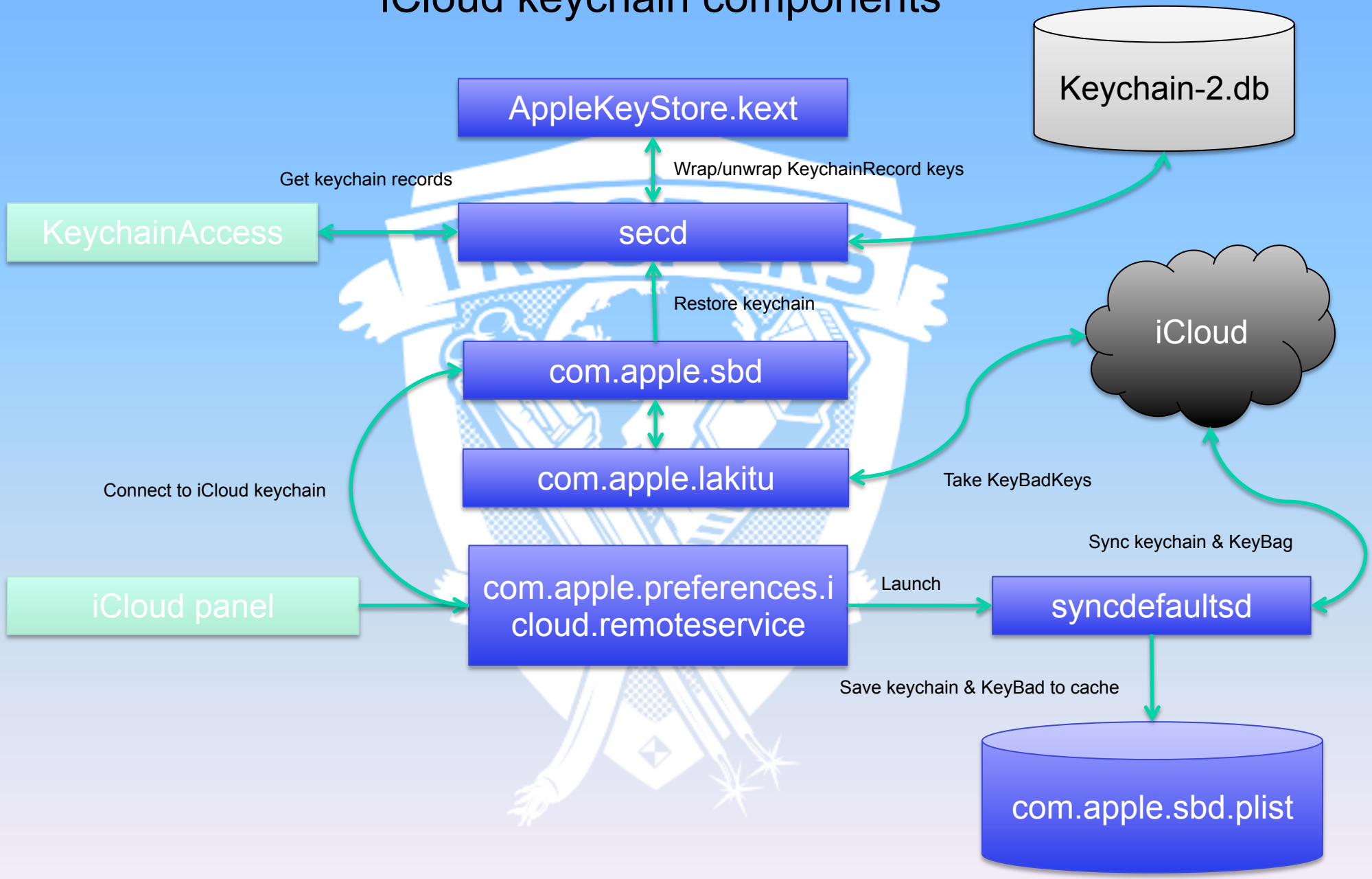
iCloud keychain (cont'd)







iCloud keychain components



iCloud keychain components

com.apple.preferences.icloud.remoteservice
process that interacts with iCloud control

com.apple.sbd
(daemon) caching and restoring keychain, get notifications

com.apple.lakitu
(daemon) talks to iCloud (get requests from com.apple.sbd, make queries to iCloud, get and decrypt responses, passes them back to com.apple.sbd)

secd
(daemon) caching and restoring keychain, get notifications

AppleKeyStore.kext
driver to interact with KVS (key-value storage)

Security.framework
functions to restore keychain, save it to SQLite database, send notifications (e.g. to Keychain access)

libcorecrypto.dylib
encryption/decryption

iCloud keychain retrieval

GetAccountSettings

POST https://setup.icloud.com/setup/get_account_settings HTTP/1.1
Host: setup.icloud.com
Authorization: Basic MTc3MzgyNT … INLy9TbkU9
[…]

Authorization = Base64(“apple_id” + “:” + “password”)
Get mmeAuthToken in return
X-MobileMe-AuthToken = Base64(“dsid” + “:” + “mmeAuthToken”)

Sync (get Keychain and KeyBag) - query

POST https://p18-keyvalueservice.icloud.com/sync HTTP/1.1
Host: p18-keyvalueservice.icloud.com
uthorization: X-MobileMe-AuthToken MTc3Mzg … meWRINDg9
[...]

```
<dict>
        <key>apns-token</key>
        <data>
        D7wxUEz2av7JaSgJD6j2lyQKENzH0e4DGJzfOeLBbYA=
        </data>
        <key>apps</key>
        <array>
                <dict>
                        <key>bundle-id</key>
                        <string>com.apple.security.cloudkeychainproxy3</string>
                        <key>kvstore-id</key>
                        <string>com.apple.security.cloudkeychainproxy3</string>
                        <key>registry-version</key>
                        <string>FT=-@RU=40c72786-6f77-4190-85d8-3ae1f4df91ca@S=1286</string>
                </dict>
                <dict>
                        <key>bundle-id</key>
                        <string>com.apple.sbd</string>
                        <key>kvstore-id</key>
                        <string>com.apple.sbd3</string>
                        <key>registry-version</key>
                        <string>FT=-@RU=40c72786-6f77-4190-85d8-3ae1f4df91ca@S=1259</string>
                </dict>
        </array>
        <key>service-id</key>
        <string>iOS</string>
</dict>
```

Sync (get Keychain and KeyBag) - response

```
<dict>
    <key>apps</key>
    <array>
      <dict>
        <key>kvstore-id</key>
        <string>com.apple.security.cloudkeychainproxy3</string>
        <key>keys</key>
        <array>
          <dict>
            <key>data</key>
            <data>AYYkF93rOBg … AABVag==</data>
            <key>name</key>
            <string>com.apple.securebackup.record</string>
          </dict>
        </array>
        <key>bundle-id</key>
        <string>com.apple.sbd</string>
      </dict>
    </array>
    <key>timestamp</key>
    <integer>1384690786479</integer>
  </dict>
```

SMS challenge?

POST https://p18-escrowproxy.icloud.com:443/escrowproxy/api/get_sms_targets HTTP/1.1
Authorization: X-MobileMe-AuthToken MTc3Mzgy…WkwdXM9
…

```
<dict>
        <key>command</key>
        <string>GET_SMS_TARGETS</string>
        <key>version</key>
        <integer>1</integer>
</dict>
</plist>
```

```
<dict>
        <key>success</key>
        <true/>
        <key>enrollRequiresPhoneNumber</key>
        <true/>
        <key>isHSAEnabled</key>
        <true/>
…
</dict>
```

SMS challenge (cont'd)

POST https://p15-escrowproxy.icloud.com:443/escrowproxy/api/generate_sms_challenge HTTP/1.1
Authorization: X-MobileMe-AuthToken MTY5NT…T0INS3FnQnM9
…

```
<dict>
<key>command</key>
<string>GENERATE_SMS_CHALLENGE</string>
<key>label</key>
<string>com.apple.securebackup.record</string>
<key>version</key>
<integer>1</integer>
</dict>

<dict>
<key>version</key>
<integer>1</integer>
<key>success</key>
<true/>
<key>smsChallengeLength</key>
<integer>6</integer>
<key>message</key>
<string>Success</string>
</dict>
```

Get KeyBagKey – srp_init

POST https://p18-escrowproxy.icloud.com:443/escrowproxy/api/srp_init HTTP/1.1
Host: p18-escrowproxy.icloud.com:443

```
<dict>
        <key>blob</key>
        <string>dSyhi0M/…CQ==</string>
        <key>command</key>
        <string>SRP_INIT</string>
        <key>label</key>
        <string>com.apple.securebackup.record</string>
        <key>phoneNumberToken</key>
        <string>9400-8433-6132-4835-3839</string>
        <key>version</key>
        <integer>1</integer>
        <key>smsChallengeCode</key>
        <string>code</string>
</dict>
```

HTTP/1.1 200 OK
[...]

```
<plist version="1.0">
        <dict>
                <key>respBlob</key>
                <string>AAABiAAA…638rrzw8=</string>
                <key>dsid</key>
                <string>1773825601</string>
        </dict>
</plist>
```

Keychain recovery (get KeyBagKey)

POST https://p18-escrowproxy.icloud.com:443/escrowproxy/api/recover HTTP/1.1
Host: p18-escrowproxy.icloud.com:443
[...]

```
<dict>
        <key>blob</key>
        <string>AAAAYAAA    …    +m8</string>
        <key>command</key>
        <string>RECOVER</string>
        <key>version</key>
        <integer>1</integer>
</dict>
```

HTTP/1.1 200 OK
[...]

```
        <dict>
                <key>respBlob</key>
                <string>AAADKAA…1FHUaEwbQ==</string>
        </dict>
```

No prompt whether to allow the new device to use the keychain!!

Keychain decryption

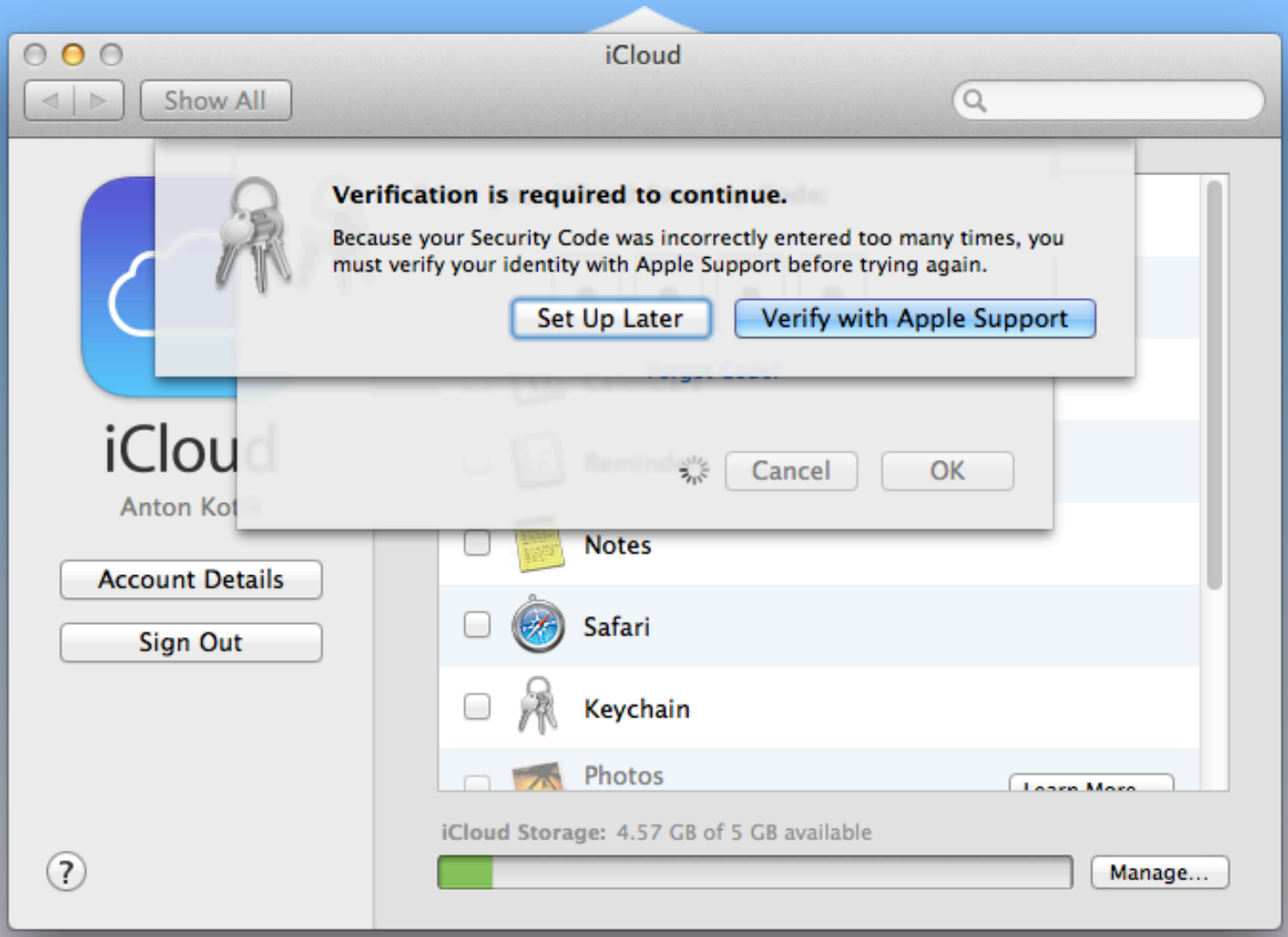
- *genp* class: Wi-Fi passwords, credit card data
- *inet* class: saved Safari passwords and autocomplete fields

- Version (4 bytes)
- KeyClass (4 bytes): index in CLASS_KEY array
- KeyDataSize (4 bytes)
- KeyData (0x48 bytes)
  - RecordPublicKey (0x20 bytes)
  - WrappedKey (0x28 bytes)
- CryptedData

- SharedKey = curve25519(CLASS_KEY.privateKey, RecordPublicKey)
  md = sha256("\x00\x00\x00\x01" + SharedKey + RecordPublicKey + CLASS_KEY.publicKey)
  AES_unwrap_key(md)
- gcm_init_key
- gcm_decrypt_message

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00000000	03	00	00	00	06	00	00	00	48	00	00	00	8C	B0	01	CE
00000010	F5	80	CD	22	77	5F	83	CD	C7	9B	4A	91	0E	47	B2	3E
00000020	DE	A4	2B	1D	26	A2	17	25	B2	CF	A1	20	1A	CA	A9	3A
00000030	BB	4B	00	F4	7E	0E	7D	A4	7E	5F	07	4D	C2	4A	D9	31
00000040	3D	D1	2E	D6	DC	C0	31	62	4C	56	99	5B	43	5C	F6	B7
00000050	61	8E	6B	A6	79	B2	BF	D9	68	74	FA	18	31	89	26	49

iCloud keychain – no brute-force available



Apple iCloud: conclusion

- Balance between security, privacy and convenience
- iCloud security risks
- Use additional encryption
- Better 2FA implementation
- Need further work
  - My Photo Stream
  - Photo Sharing
  - 3rd party apps data
  - Back To My Mac
  - Touch ID (iPhone 5S)

BlackBerry backups

Old format:

- IPD files (all databases in a single container)
- BBB files (in fact, ZIP archives with several IPDs, one database per IPD)

New format:

- Unencrypted BBB-QNX (three .tar files inside); for PlayBook with firmware <2.0
- Encrypted BBB-QNX (all .tar files are encrypted); for BB OS 10 (created w/ BlackBerry Link)

For old formats - simple password protection:

- Encryption: AES-256
- Password verification:

  BlackBerry Desktop Software 5: pbkdf2 (1) - *yes, just one iteration*

  BlackBerry Desktop Software 6: pbkdf2 (20,000)

BB 10 backups

- mounting QNX6 partitions
- backup encryption: AES-256
- authentication/verification: HMAC-SHA1
- backup.cgi:backuparch
- backup.cgi:scramble

  bbid (BlackBerry ID)

  (libbbid.so:bbid_profile_get_user_properties(urn:bbid:username)

  qbek

  (libbbid.so:bbid_profile_get_user_properties(urn:bbid:backupandrestorekey)

  cache storage: /accounts/<id>/sys/bbid/keyCache

  if not found: request to BB Olympia Service (blackberryid.blackberry.com)

BlackBerry token service

- request: bbid, password, pin, salt (client's entropy)
- response:

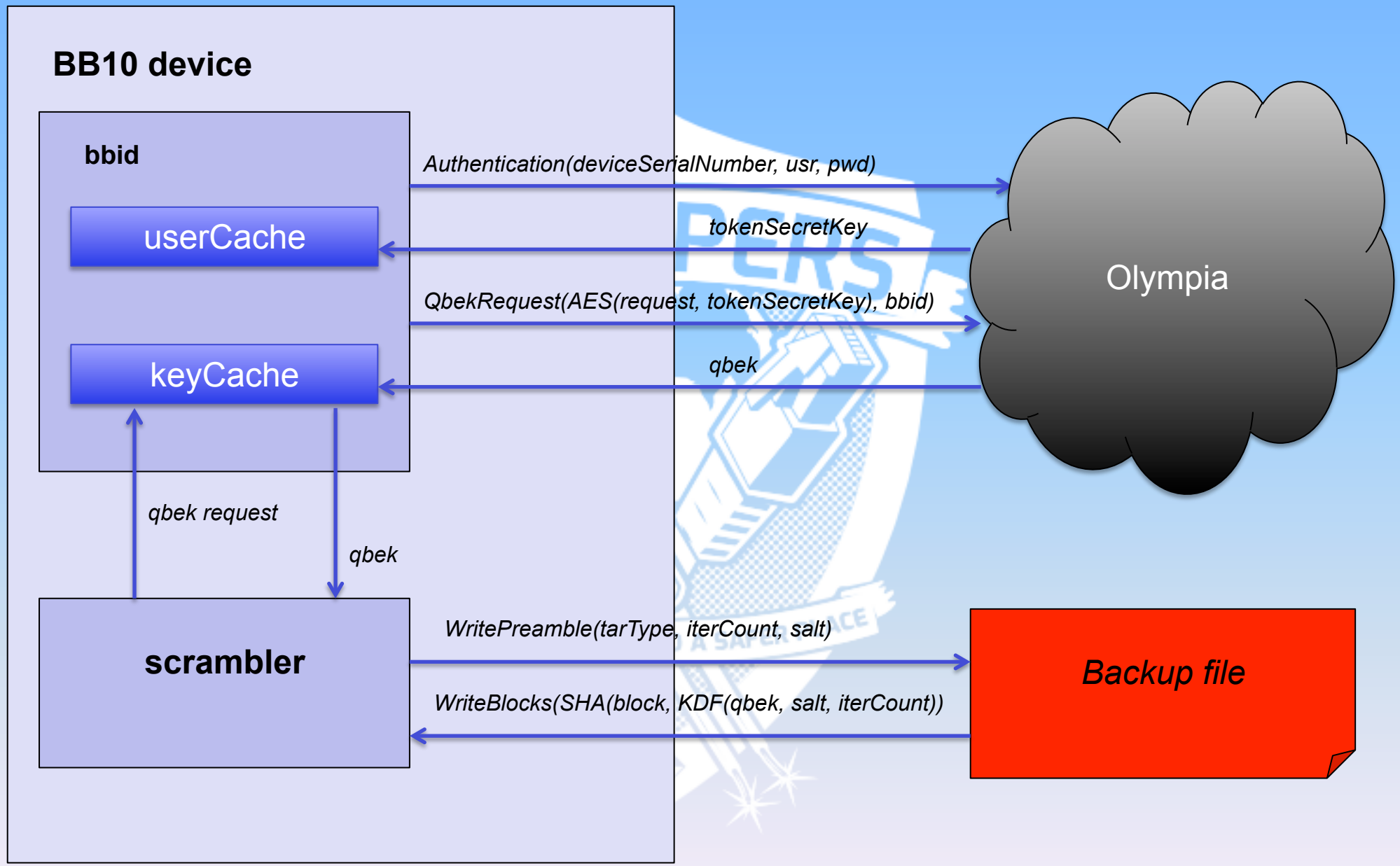
  Hct=1379081439336&st=1379168703336&se=3296-4542-6332-2304-3821&et=1381673439336&fn=John&ln=Doe&nn=johndoe-59094&un=john.doe%40gmail.com&ec=AcDGzWbVM12nd0BigqIfJYw%3D&em=john.doe%40gmail.com&at=AQ:AQ:zTh0_L5BwTuZf0w0L2CYVGmMyrzSbs7OszPBq72NIYY:ibKt2ZKGOsAjODk6lITmQA:asSsJMYRzS8Tf2IMQY44_HiCDaWzCBRwQj68XDDH0z6Qhp7gCXuKqSk6_v4KTQ8pWMtpVriBNBWO4t2Ig879MY_Oro2upCzw32EmCgAKapUPGTIeAlKeo3kr13v-Td2IpWU0b3kQJVJsTMz9GBjG29RFkcxw-039ksxUJYnDxkCrgbrAwVFpw5Pg5XmAZxtA

  - se - server entropy
  - at - authentication token
  - ec - user ID for BB cloud services (saved to /dev/rpmb/BBID_BDEK)
  - at (creation time), st (server time), et (expiry time)
  - further reuests: RST (Request Secure Token) with token type and sevrice name

to get qbek:

- get authentication token
- get BBIDAuthN_1 token for urn:bbid:v1:olympia)
- send request for authzo:qbek token
- register device on BB server (using authzo:qbek token)
- get request on backupAndRestoreKey info (two IDs)
- get janusUrl by request to kronos.bbprotect.blackberry.com
- get qbek from %janusUrl%/FlashGetFile

- **Valid device PIN should be supplied**
- **qbek depends on BlackBerry ID only**


BB10 device
bbid
userCache
keyCache
scrambler
Authentication(deviceSerialNumber, usr, pwd)
tokenSecretKey
QbekRequest(AES(request, tokenSecretKey), bbid)
qbek
Olympia
qbek request
qbek
WritePreamble(tarType, iterCount, salt)
WriteBlocks(SHA(block, KDF(qbek, salt, iterCount))
Backup file

BlackBerry 10 backup: what is inside

app.tar (application data)

sys.pim.attachments.card.gYABgLcka3eBcb.4E7SWEroZgCA
sys.pim.calendar.gYABgG0xvpxP1jARa6DD5o.VL8A
sys.pim.contacts.gYABgGsAOuzqCT1fu5Zx4sqrJdY
sys.pim.messages.gYABgJ8jn83Ok_NEWYplPYozt5w

com.rim.bb.app.facebook.gYABgDLo0nc9AhDgv2JAPixdyvQ\appdata\data\fb-messages.db
com.rim.bb.app.facebook.gYABgDLo0nc9AhDgv2JAPixdyvQ\appdata\data\mail.db

com.whatsapp.WhatsApp.gYABgD934jlePGCrd74r6jbZ7jk\appdata\data\contactStore.db
com.whatsapp.WhatsApp.gYABgD934jlePGCrd74r6jbZ7jk\appdata\data\messageStore.db

sys.bbm.gYABgLOJBR2Vz7FzS.kdgJchuag\appdata\data\bbgroups.db
sys.bbm.gYABgLOJBR2Vz7FzS.kdgJchuag\appdata\data\master.db

media.tar (photos and videos)

settings.tar (varios settings)

var\etc\netsecure\vpn_pps.conf
var\etc\netsecure\wpa_pps.conf

pps\services*

pps\system\bookmarks

Special thanks to our team:

Researches

- Anton K.
- Sergey P.
- Denis B.

Developers

- Alina P.
- Oleg K.

Testers

- Artem G.
- Oxana B.

Managers

- Anna S.
- Vladimir S.

I could not have done it without you! ;)

Thank you!

Vladimir Katalov, ElcomSoft Co. Ltd.
(twitter: @vkatalov)

http://www.elcomsoft.com
http://blog.crackpassword.com
Facebook: ElcomSoft
Twitter: @elcomsoft