

My activities related to smart card and Free Software (as in free speech).

Friday, March 31, 2017

To continue the list of PC/SC wrappers initiated in 2010 with "PC/SC sample in different languages" I now present a new sample in **Objective-C** using the Apple Crypto Token Kit API.

I already proposed a sample code in Objective-C in "[PCSC sample in Objective-C](#)". This code used the [asynchronous version of sendIn](#). The API is:

```
- (void)sendIns:(UInt8)ins
    p1:(UInt8)p1
    p2:(UInt8)p2
    data:(NSData *)requestData
    le:(NSNumber *)le
    reply:(void (^)(NSData *replyData, UInt16 sw, NSError *error))reply;
```

The method returns immediately and a callback `reply` block is executed when the card response is received.

We will now use the **synchronous version of sendInS**. The API is:

```
- (NSData *)sendIns:(UInt8)ins
    p1:(UInt8)p1
    p2:(UInt8)p2
    data:(NSData *)requestData
    le:(NSNumber *)le
    sw:(UInt16 *)sw
    error:(NSError * _Nullable *)error;
```

In Yosemite (Mac OS X 10.10) Apple introduced a new API to access smart cards. See [OS X Yosemite and smart cards status](#).

This API is not a wrapper above PC/SC. It is the native API to be used on macOS. You do not need to install it, it comes with the OS.

Since PC/SC is not used here the blog title may be misleading. So I used " " around PC/SC this time.

Create a new Cocoa application in Xcode. You need to enable the App Sandbox and add/set the `com.apple.security.smartcard` entitlement to yes.

My sample HelloWorld application does not use Cocoa. It is a text only application.

```
#import <CryptoTokenKit/CryptoTokenKit.h>

int main(int argc, const char * argv[])
{
    TKSmartCardSlotManager * mngr;
    mngr = [TKSmartCardSlotManager defaultManager];

    // Use the first reader/slot found
    NSString *slotName = (NSString *)mngr.slotNames[0];
    NSLog(@"slotName: %@", slotName);

    dispatch_semaphore_t sem = dispatch_semaphore_create(0);

    // connect to the slot
    [mngr getSlotWithName:slotName reply:^(TKSmartCardSlot *slot)
    {
        // connect to the card

        TKSmartCard *card = [slot makeSmartCard];
        if (nil == card)
        {
            NSLog(@"No card found");

            // signals end of getSlotWithName block
            dispatch_semaphore_signal(sem);

            return;
        }

        // begin a session
        [card beginSessionWithReply:^(BOOL success, NSError *error)
        {
            if (success)
            {
                NSData *response;
                UInt16 sw;
                NSString *newString;

                // explicitly set the CLA byte even if 0 is already the default value
                card.cla = 0x00;
            }
        }];
    }];
}
```

Google+ Badge

Ludovic Rousseau blog



Blog Archive

▼ 2017 (33)

- ▶ November (1)
- ▶ October (3)
- ▶ September (4)
- ▶ August (1)
- ▶ June (2)
- ▶ May (5)
- ▶ April (1)

▼ March (6)

"PC/SC" sample in Objective-C
(synchronous)

Gemalto IDBridge K30, K50,
CT30 and Zero Length Pa...

PC/SC sample in Rust

ATR statistics: TC2 - Specific to T=0

PC/SC sample in Smart Card
Connector on
Chromebook...

macOS Sierra bug:
SCardTransmit() silently
truncat...

- ▶ February (4)
- ▶ January (6)

- ▶ 2016 (49)
- ▶ 2015 (51)
- ▶ 2014 (61)
- ▶ 2013 (38)
- ▶ 2012 (27)
- ▶ 2011 (46)
- ▶ 2010 (55)

Search This Blog

Search

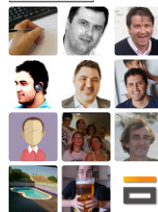
Subscribe To

- Posts
- Comments

Google+ Followers

Ludovic Rousseau b...

Follow



336 have us in circles [View all](#)

```

        // send 1st APDU
        uint8_t aid[] = {0xA0, 0x00, 0x00, 0x00, 0x62, 0x03, 0x01, 0x0C, 0x0
6, 0x01};

        NSData *data = [NSData dataWithBytes:aid length:sizeof aid];

        response = [card sendIns:0xA4 p1:0x04 p2:0x00 data:data le:nil sw:&s
w error:&error];

        if (nil == response)
        {
            NSLog(@"sendIns error: %@", error);
            goto out;
        }

        NSLog(@"Response: %@ 0x%04X", response, sw);

        // send 2nd APDU
        response = [card sendIns:0x00 p1:0x00 p2:0x00 data:nil le:@0 sw:&sw
error:&error];

        if (nil == response)
        {
            NSLog(@"sendIns error: %@", error);
            goto out;
        }

        NSLog(@"Response: %@ 0x%04X", response, sw);

        newString = [[NSString alloc] initWithData:response encoding:NSUTF8
StringEncoding];
        NSLog(@"%@", newString);

    out:

        // end the session
        [card endSession];
    }
    else
    {
        NSLog(@"Session error: %@", error);
    }

    // signals end of beginSessionWithReply block
    dispatch_semaphore_signal(sem);
    });
    });

    // wait for the asynchronous blocks to finish
    dispatch_semaphore_wait(sem, DISPATCH_TIME_FOREVER);

    return 0;
}

```

Output

```

2017-03-31 10:54:24.990581+0200 HelloWorld[19931:85555] slotName: Gemalto PC Twin Read
er
2017-03-31 10:54:25.103855+0200 HelloWorld[19931:85584] Response: <> 0x9000
2017-03-31 10:54:25.115946+0200 HelloWorld[19931:85584] Response: <48656c6c 6f20776f
d="" 726c6421=""> 0x9000
2017-03-31 10:54:25.115993+0200 HelloWorld[19931:85584] Hello world!

```

Comments

Compared to the previous Objective-C sample in "PCSC sample in Objective-C" this code has some improvements/bugs fixes:

- `[card endSession];` is called.
This is needed to close the session started by `[card beginSessionWithReply:...]`.
- The main thread is waiting for the callbacks from `[mgr getSlotWithName:...]` and `[card beginSessionWithReply:...]` to finish using a semaphore (instead of a `sleep()`).

The CryptoTokenKit API provides a `inSessionWithError:executeBlock:` to synchronously begin a session instead of using `beginSessionWithReply:` and `endSession`. But this method has some limitations/bugs and is not (yet) easy to use. I may use it in a next sample code when it will be fixed (in macOS 10.13?).

Conclusion

In general, I prefer to use synchronous calls. So the possibility to use a synchronous `sendIns:` method is nice.

Depending on your needs, the `CryptoTokenKit TKSmartCard` API offers you the choice between a synchronous or asynchronous version.



Labels: code, Mac OS X

[Newer Post](#)

[Home](#)

[Older Post](#)

Bitcoin



License: by-nc-sa



This blog by [Ludovic Rousseau](#) is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License](#).

Simple theme. Powered by [Blogger](#).