

iOS

- Bluetooth,
- iPhone, iPad iPod touch .
- , , , , CSP.
- CSP -, framework iOS, .
- 28147-89 - .
- 34.10-2001 - .
- VKO 34.10-2001 (RFC 4753) - 28147 34.10-2001.
- 28147 , Bluetooth.
- Secure Messaging -, , .
- ()- 34.10-2001.
- .
- , , (,)

34.10-2001, , VKO 34.10-2001.

,, , CSP:

```
@interface CProReader : NSObject
@property (assign, readwrite) NSString* nickname;
@property (assign, readwrite) NSString* name;
@property (assign, readwrite) NSString* media;
@property (assign, readwrite) uint8_t flags;
-(void)dealloc;
-(CProReader*) initWithData:(uint8_t*)dataPtr;
@end

@implementation CProReader

@synthesize name;
@synthesize nickname;
@synthesize media;
@synthesize flags;
```

```

-(CProReader*) init {
    [super init];
    self.name = nil;
    self.nickname = nil;
    self.media = nil;
    return self;
}

-(CProReader*) initWithData: (uint8_t*)dataPtr {
    [super init];
    self.nickname = [[[NSString alloc] initWithBytes:dataPtr length:strlen((char*)dataPtr) encoding:
    NSUTF8StringEncoding] autorelease];
    dataPtr+=1+[self.nickname length];
    self.name = [[[NSString alloc] initWithBytes:dataPtr length:strlen((char*)dataPtr) encoding:
    NSUTF8StringEncoding] autorelease];
    dataPtr+=1+[self.name length];
    self.media = [[[NSString alloc] initWithBytes:dataPtr length:strlen((char*)dataPtr) encoding:
    NSUTF8StringEncoding] autorelease];
    dataPtr+=1+[self.name length];
    self.flags = *dataPtr;
    return self;
}

-(void)dealloc {
    [super dealloc];
}
@end

static const int kGostProvType = 75;

NSArray* getReaderList()
{
    NSMutableArray* readerList = nil;

    DWORD error = ERROR_SUCCESS;
    HCRYPTPROV hCryptProv = 0;
    CSP_BOOL bResult = 0;
    DWORD dwLen = 0;

    bResult = CryptAcquireContext(&hCryptProv, NULL, NULL, kGostProvType, CRYPT_VERIFYCONTEXT);
    if (!bResult) {
        error = CSP_GetLastError();
        NSLog(@"CryptAcquireContext(CRYPT_VERIFYCONTEXT): %x\n", error);
    }

    if(0 == hCryptProv) {
        NSLog(@"Invalid HCRYPTPROV");
        return nil;
    }

    BYTE cryptFirst = CRYPT_FIRST;

    for (;1;) {

        CSP_SetLastError(ERROR_SUCCESS);
        bResult = CryptGetProvParam(hCryptProv, PP_ENUMREADERS, NULL, &dwLen, CRYPT_MEDIA | cryptFirst);
        error = CSP_GetLastError();
        if (error == ERROR_NO_MORE_ITEMS)
            break;
        if (!bResult)
        {
            printf("CryptGetProvParam(PP_ENUMREADERS, LEN): %x\n", error);
            break;
        }
    }

    NSMutableData* data = [[[NSMutableData alloc] initWithCapacity:dwLen] autorelease];

    CSP_SetLastError(ERROR_SUCCESS);
    bResult = CryptGetProvParam(hCryptProv, PP_ENUMREADERS, (BYTE*)[data bytes], &dwLen, CRYPT_MEDIA |

```

```

cryptFirst);
    cryptFirst = 0;
    error = CSP_GetLastError();
    if (error == ERROR_NO_MORE_ITEMS)
        break;
    if (!bResult)
    {
        printf("CryptGetProvParam(PP_ENUMREADERS, NAME): %x\n", error);
        break;
    }

    BYTE* dataPtr = (BYTE*)[data bytes];
    CProReader* reader = [[[CProReader alloc] initWithData:dataPtr] autorelease];

    if (nil == readerList) {
        readerList = [[NSMutableArray new] autorelease];
    }

    [readerList addObject: reader];
}
return readerList;
}

```

```

getReaderList() CProReader name=@"Aktiv Rutoken ECP BT XXXXXXXX" media @"rutoken_ecp_YYYYYYY", , . . .
CProReader name=@"Aktiv Rutoken ECP BT XXXXXXXX" media @"NO_MEDIA", . hex(XXXXXXXX) pkcs11 :

```

```

#define SLOTS_MAX_COUNT 100
@interface PKCS11 : NSObject
+(long)setActivationPassword:(NSString*)password forReader:(NSString*)reader;
@end

@implementation PKCS11
+(long)setActivationPassword:(NSString *)password forReader:(NSString *)reader
{
    CK_FUNCTION_LIST_PTR          pFunctionList          = NULL_PTR; // PKCS#11, CK_FUNCTION_LIST
    CK_FUNCTION_LIST_EXTENDED_PTR  pFunctionListEx       = NULL_PTR; // PKCS#11,
    CK_FUNCTION_LIST_EXTENDED
    CK_RV                          rv                   = CKR_OK; //
    CK_ULONG                       ulSlotCount         = SLOTS_MAX_COUNT;
    CK_SLOT_ID                      slots[SLOTS_MAX_COUNT];

    while(true) {
        /******
        * 1:
        *          PKCS#11.
        *          *
        *****/
        printf("Getting function list");
        rv = C_GetFunctionList(&pFunctionList);
        if (rv != CKR_OK)
        {
            printf(" -> Failed\n");
            break;
        }
        printf(" -> OK\n");

        /******
        * 2:
        *          PKCS#11.
        *          *
        *****/
        printf("Getting extended function list");
        rv = C_EX_GetFunctionListExtended(&pFunctionListEx);
        if (rv != CKR_OK)
        {
            printf(" -> Failed\n");
            break;
        }
        printf(" -> OK\n");

        /******

```

```

* 3: . *
*****/
printf("Initializing library");
rv = pFunctionList->C_Initialize(NULL_PTR);
if (rv != CKR_OK)
{
    printf(" -> Failed\n");
    break;
}
printf(" -> OK\n");

/*****
* 4: *
*****/
printf("Getting slots");
rv = pFunctionList->C_GetSlotList(CK_TRUE, slots, &ulSlotCount);
if (rv != CKR_OK)
{
    printf(" -> Failed\n");
    break;
}
printf(" -> OK\n");

for (size_t i = 0 ; i < ulSlotCount; ++i) {
/*****
* 5: *
*****/
printf("Getting tokeninfo\n");
CK_TOKEN_INFO tokenInfo;
rv = pFunctionList->C_GetTokenInfo(slots[i], &tokenInfo);
if (rv != CKR_OK)
{
    printf(" -> Failed\n");
    continue;
}
printf(" -> OK\n");

/*****
* 6: , *
*****/
printf("Try open session\n");
CK_SESSION_HANDLE hSession;
rv = pFunctionList->C_OpenSession( slots[i],
                                CKF_SERIAL_SESSION | CKF_RW_SESSION,
                                NULL_PTR,
                                NULL_PTR,
                                &hSession);

if(rv == CKR_OK) {
    printf("Token is valid\n");
    rv = pFunctionList->C_CloseSession(hSession);
    continue;
}
printf(" -> OK\n");

/*****
* 7: *
*****/
int j = 0;
for(j = 0; j < sizeof(tokenInfo.serialNumber) && tokenInfo.serialNumber[j] != ' '; ++j);
if(j < sizeof(tokenInfo.serialNumber))
    tokenInfo.serialNumber[j] = '\0';
else
    continue; // rutoken serial is less than 16 charecters

NSString* hexSerial = [NSString stringWithCString:tokenInfo.serialNumber encoding:
NSUTF8StringEncoding];
size_t serialInt;
NSScanner *scanner = [NSScanner scannerWithString:hexSerial];
[scanner scanHexInt:&serialInt];
NSString* serial = [NSString stringWithFormat:@"%zd", serialInt];

```

```

// checking if it is the same rutoken
if([reader rangeOfString:serial].location != NSNotFound) {
    /*****
     * 8:
     *****/
    printf("Setting activation password");
    rv = pFunctionListEx->C_EX_SetActivationPassword(slots[i], (CK_UTF8CHAR_PTR)[password
cStringUsingEncoding:NSUTF8StringEncoding]);
    if (rv != CKR_OK)
    {
        printf(" -> Failed\n");
        continue;
    }
    printf(" -> OK\n");
}

/*****
 * 9: ,
 *****/
printf("Check session now can be opened\n");
rv = pFunctionList->C_OpenSession( slots[i],
                                CKF_SERIAL_SESSION | CKF_RW_SESSION,
                                NULL_PTR,
                                NULL_PTR,
                                &hSession);

if(rv != CKR_OK) {
    printf("Failed to open session\n");
    printf("Setting activation password seems to have failed\n");
    continue;
}
printf(" -> OK\n");
pFunctionList->C_CloseSession(hSession);
}
break;
}

if (pFunctionList)
{
    /*****
     *
     *****/
    int rvTemp = CKR_OK;
    printf(" Finalizing library");
    rvTemp = pFunctionList->C_Finalize(NULL_PTR);
    if (rvTemp != CKR_OK)
        printf(" -> Failed\n");
    else
        printf(" -> OK\n");
    pFunctionList = NULL_PTR;
}
return rv;
}
@end

```

:

- (C_GetSlotList, C_GetTokenInfo);
- , , (C_OpenSession CKR_FUNCTION_NOT_SUPPORTED = 0x54);
- _EX_SetActivationPassword(CK_SLOT_ID slotID, CK_UTF8CHAR_PTR password);
- ;
- C_CloseSession pkcs11 C_Finalize.

getReaderList() CProReader name=@"Aktiv Rutoken ECP BT XXXXXXXX" media @"rutoken_ecp_YYYYYYY", CSP

.

(), .

(). ().

, . " " , . , .

, :

- Info.plist (Supported external accessory protocols) "com.aktivco.rutokenecp"

```
<?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>UISupportedExternalAccessoryProtocols</key>
<array>
...
<string>com.aktivco.rutokenecp</string>
...
</array>
...
</dict>
</plist>
```

• , , :

- Entitlements.plist :

Entitlements.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>application-identifier</key>
<string>YYYYYYYYYY.my.company.app-identifier</string>
<key>get-task-allow</key>
<true/>
<key>keychain-access-groups</key>
<array>
<string>YYYYYYYYYY.my.company.keychainAccessGroupWithRutokenStringInIt</string>
</array>
</dict>
</plist>
```

my.company.app-identifier – , YYYYYYYYYY – Team ID (<https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/AppDistributionGuide/MaintainingProfiles/MaintainingProfiles.html>), Bundle Seed ID App ID prefix (<https://developer.apple.com/library/mac/documentation/general/conceptual/devpedia-cocoacore/AppID.html>),

my.company.keychainAccessGroupWithRutokenStringInIt -- (https://developer.apple.com/library/ios/documentation/Security/Reference/keychainservices/Reference/reference.html#apple_ref/doc/uid/TP30000898-CH1g-SW2), – "rutoken".

- entitlements Entitlements1.plist (), keychain-access-groups YYYYYYYYYY.my.company.keychainAccessGroupWithRutokenStringInIt.

Entitlements.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>keychain-access-groups</key>
    <array>
        ...
        <string>YYYYYYYYYY.my.company.keychainAccessGroupWithRutokenStringInIt</string>
        ...
    </array>
    ...
</dict>
</plist>
```

- Xcode "Code Signing"->"Code Signing Entitlements" Entitlements1.plist, YYYYYYYYYY.my.company.keychainAccessGroupWithRutokenStringInIt Entitlements.plist Entitlements1.plist Apple Developer ID.
- Entitlements.plist , Entitlements1.plist entitlements (Entitlements1.plist keychain-access-groups , Entitlements.plist) – ;
- Entitlements.plist Entitlements1.plist entitlements (Entitlements1.plist keychain-access-groups) – , .
- Entitlements.plist , Entitlements1.plist entitlements, Entitlements1.plist keychain-access-groups, "YYYYYYYYYY.my.company.someKeychainAccessGroup",- , , Entitlements1.plist entitlements, Entitlements1.plist keychain-access-groups, "YYYYYYYYYY.my.company.someKeychainAccessGroup".
- :
 - RDRRtSupCp.framework – Rutoken ECP BT ;
 - RtPKCS11ECP.framework – , PKCS#11;
 - RtPcsc.framework – PCSC-;
 - Security.framework, ExternalAccessory.Framework, Foundation.framework.

- 1)
 - 1.1.1) Bluetooth (, Bluetooth , 1.1.1-1.1.3)
 - 1.1.2) Bluetooth (iOS7) Bluetooth
 - 1.1.3) iOS6
 - 1.2.1) (,)
 - 1.2.2) "" Bluetooth Bluetooth " ".
 - 1.2.3)
 - 1.2.4)
- 2) , **CryptGetProvParam(PP_ENUMREADERS)** szMedia="NO_MEDIA" CRYPT_ENUMREADER_INFO_MEDIA (http://cpdn.cryptopro.ru/content/csp36/html/struct__c_r_y_p_t_e_n_u_m_r_e_a_d_e_r__i_n_f_o__m_e_d_i_a.html), . , , .
- 3) pkcs11 CKR_OK C_OpenSession
- 4)
- 5)

- 1)
 - 1.1.1) Bluetooth (, Bluetooth , 1.1.1-1.1.3)
 - 1.1.2) Bluetooth (iOS7) Bluetooth

1.1.3) iOS6

1.2.1) (,)

1.2.2)

2) CryptGetProvParam(PP_ENUMREADERS) szMedia="rutoken_ecp_XXXXXXXX",

3)