Защищённый канал на iOS

Терминология

Токен - Рутокен ЭЦП Bluetooth, поддерживающий обмен по защищённому каналу. Т. основан на Рутокен ЭЦП, содержит в себе реализацию российских криптографических алгоритмов шифрования и электронной подписи.

Мобильное устройство - iPhone, iPad или iPod touch на котором установлено приложение.

Приложение - программа, установленная на мобильном устройстве, включающая в себя криптопровайдер, например, КриптоПро CSP.

КриптоПро CSP - криптопровайдер, специальный framework для iOS, содержащий в себе реализацию российских криптографических алгоритмов шифрования и электронной подписи.

ГОСТ 28147-89 - российский национальный стандарт шифрования.

ГОСТ Р 34.10-2001 - российский национальный стандарт электронной подписи.

VKO ГОСТ Р 34.10-2001 (RFC 4753) - технология выработки секретных симметричных ключей ГОСТ 28147 на основе одного закрытого и одного открытого ключей ГОСТ Р 34.10-2001.

Защищённый канал - зашифрованный по ГОСТ 28147 поток данных, пересылаемый между устройством и Токеном по радиоканалу Bluetooth.

Secure Messaging - технология, по которой работает защищённый канал, гарантирующая секретность и сохранность пересылаемых в недоверенной среде данных.

Ключевая пара (асимметричная ключевая пара) - закрытый и открытый ключ ГОСТ 34.10-2001.

Первоначальная инициализация канала - процесс генерации ключевых пар на Токене и на мобильном устройстве и процесс обмена открытыми ключами между Токеном и мобильным устройством.

Пароль инициализации - специальный пароль, записанный на карточке, которая может входить в комплект поставки Токена. Без знания пароля инициализации приложение не может произвести первоначальную инициализацию защищённого канала с Токеном. (если пароля нет, то его можно сгенерировать через Панель управления Рутокен по инструкции Политики безопасности и шифрование канала)

Общая схема

Шифрование в защищённом канале происходит на короткоживущих сессионных ключах, которые в процессе работы канала периодически меняются.

Сессионные ключи вырабатываются из долговременных асимметричных ключевых пар ГОСТ34.10-2001, которые находятся на мобильном устройстве и на Токене, по алгоритму VKO ГОСТ Р 34.10-2001.

Долговременные ключевые пары вырабатываются на Токене и на мобильном устройстве при первоначальной инициализации защищённого канала.

Для того чтобы обе стороны могли генерировать один и тот же симметричный сессионный ключ, они сперва должны обменяться открытыми ключами.

Обмен открытыми ключами между Токеном и устройством происходит на одном общем симметричном ключе, которые должен присутствовать как на Токене, так и на мобильном устройстве. В Токене этот общий ключ прошивается на заводе, его нельзя просмотреть или изменить.

Так как заранее неизвестно, с каким именно мобильным устройством будет работать Токен, на мобильное устройство этот общий ключ должен попасть при участии пользователя, то есть введён через графический интерфейс.

Токены с защитой канала комплектуются специальными карточками, на которых записана некоторая последовательность символов - "пароль инициализации". Этот пароль должен быть введен на мобильном устройстве через графический интерфейс приложения.

С помощью криптографических преобразований на мобильном устройстве этот пароль преобразуется в общий ключ, который используется для инициализации защищенного канала.

Путём обмена зашифрованными на общем ключе сообщениями происходит обмен открытыми ключами; если обмен был успешен, открытые ключи сохраняются на мобильном устройстве и на Токене, а общий ключ удаляется, так как больше не нужен.

Когда обмен открытыми ключами между мобильным устройством и Токеном произошел, созданы все условия для того чтобы создание защищённого канала между мобильным устройством и Токеном было успешным.

Важно понимать, что, хотя первоначальная инициализация защищённого канала требует дополнительных действий, после её прохождения защищённый канал начинает работать совершенно прозрачно, как будто его и нет.

Алгоритм первоначальной инициализации защищенного канала

В первую очередь, для проверки, что защищенный канал еще не инициализирован, необходимо перечислить список подключенных доступных КриптоПро CSP считывателей:

```
@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;
}
```

В случае если в возвращенном функцией getReaderList() массиве считывателей у объекта CProReader с именем name=@"Aktiv Rutoken ECP BT XXXXXXXX" поле media имеет значение @"rutoken_ecp_YYYYYYY", делать больше ничего не надо, всё уже готово для работы. Первоначальная инициализация защищенного канала уже произведена.

Если у объекта CProReader с именем name=@"Aktiv Rutoken ECP BT XXXXXXXX" поле media имеет значение @"NO_MEDIA", необходимо произвести инициализацию защищенного канала. Для этого к Токену с серийным номером hex(XXXXXXXX) следует обратиться к Токену через интерфейс pkcs11 и произвести инициализацию:

```
#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
```

В предложенном исходном коде для первоначальной инициализации защищенного канала производятся следующие действия:

- поиск токена с заданным серийным номером (**C_GetSlotList**, **C_GetTokenInfo**);
- проверка, что сессия на токене, для которого не была произведена первоначальная инициализация защищенного канала, не открывается (функция **C_OpenSession** возвращает ошибку **CKR_FUNCTION_NOT_SUPPORTED = 0x54**);
- инициализация защищенного канала посредством вызова функции **C_EX_SetActivationPassword(CK_SLOT_ID slotID, CK_UTF8CHAR_PTR password);**
- проверка, что сессия на токене после инициализация защищенного канала открывается успешно;
- закрыти сессии функцией **C_CloseSession** и завершение работы с интерфейсом pkcs11 с помощью функции **C_Finalize.**

После первоначальной инициализации защищенного канала в возвращенном функцией getReaderList() массиве считывателей у объекта CProReader с именем name=@"Aktiv Rutoken ECP BT XXXXXXXX" поле media имеет будет значение @"rutoken_ecp_YYYYYYY", и дальнейшее взаимодействие с Токеном через КриптоПро CSP на мобильном устройстве ничем не отличается от взаимодействия с Токеном на других платформах.

Условия работы защищённого канала

Защищённый канал устанавливается между парой конкретных экземпляров приложения и Токена.

Если заменён хотя бы один из членов пары (приложение переустановлено или заменён Токен), защищённый канал потребует переинициализации.

Одно приложение может иметь возможность проинициализировать защищённый канал с множеством Токенов (в приложении могут храниться открытые ключи от нескольких токенов). Также один Токен может имеет возможность проинициализировать защищённый канал с множеством приложений (токен может хранить несколько открытых ключей приложений).

Несколько приложений одного и того же разработчика могут пользоваться одной и той же ключевой информацией, которая генерируется при инициализации защищённого канала одним из приложений. Это позволяет уменьшить количество ввода "пароля инициализации" пользователем до одного раза, даже если он устанавливает не одно приложение, а целый пакет. Пароль инициализации может быть введён в любом приложении и другие приложения при этом будут вести себя так, как будто в них он уже введен.

Требования к приложению

На приложение, обращающееся к Токену накладываются следующие требования:

- В Info.plist приложения должен быть указан протокол поддерживаемых внешних устройств (Supported external accessory protocols) со значением "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>
```

- В случае, если требуется обеспечить возможность одной и той же ключевой информации в нескольких приложениях разработчика, необходимо обеспечить выполнение следующих требований:
    - В ресурсы приложения должен быть добавлен файл Entitlements.plist следующего содержания:

      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>
      ```

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

      my.company.keychainAccessGroupWithRutokenStringInIt -- идентификатор хранилища ключей (подробнее https://developer.apple.com/library/ios/documentation/Security/Reference/keychainservices/Reference/reference.html#//apple_ref/doc/uid/TP30000898-CH1g-SW2), в которую будут сохраняться долговременные ключи на мобильном устройстве – данная строка должна содержать в себе подстроку "rutoken".
    - Приложение должно быть подписано с указанием в качестве источника entitlements файла Entitlements1.plist (имя может быть использовано любое), в котором должен содержаться параметр keychain-access-groups со значением `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>
```

Для этого в Xcode в настройке параметров сборки приложения в параметре "Code Signing"->"Code Signing Entitlements" необходимо указать путь до Entitlements1.plist.

- Для того чтобы воспользоваться возможностью ввода пароля инициализации один раз для всех приложений разработчика, строка YYYYYYYYYY.my.company.keychainAccessGroupWithRutokenStringInIt в файлах Entitlements.plist и Entitlements1.plist этих приложений должна быть одинаковая и эти приложения должны быть подписаны одной и той же подписью Apple Developer ID.

В случае, если перечисленные требования не выполнены или выполнены не в полной мере, возможны следующие варианты:

- Корректный файл Entitlements.plist добавлен в ресурсы приложения, но приложение подписано без указания Entitlements1.plist в качестве источника entitlements (или в Entitlements1.plist не указан параметр keychain-access-groups с тем же значением, что в Entitlements.plist) – инициализация защищенного канала будет завершена с ошибкой; открытие защищенного канала невозможно.
- Файл Entitlements.plist не добавлен в ресурсы приложения и приложение подписано без указания Entitlements1.plist в качестве источника entitlements (или в Entitlements1.plist не указан параметр keychain-access-groups) – инициализация защищенного канала пройдет успешно, сохраненная ключевая информация доступна только одному приложению.
- Файл Entitlements.plist не добавлен в ресурсы приложения, и приложение подписано с указанием Entitlements1.plist в качестве источника entitlements, и в Entitlements1.plist указан параметр keychain-access-groups, в котором первым значением в массиве является строка "YYYYYYYYYY.my.company.someKeychainAccessGroup", – инициализация защищенного канала пройдет успешно, сохраненная ключевая информация будет доступна всем приложениям разработчика, которые аналогично подписаны с указанием Entitlements1.plist в качестве источника entitlements, где в Entitlements1.plist указан параметр keychain-access-groups, в котором первым значением в массиве является строка "YYYYYYYYYY.my.company.someKeychainAccessGroup".

- Приложение должно быть слинковано со следующими фреймворками:
  - RDRRtSupCp.framework – модуль поддержки считывателя Rutoken ECP BT для КриптоПро;
  - RtPKCS11ECP.framework – модуль, реализующий стандарт PKCS#11;
  - RtPcsc.framework – модуль поддержки PCSC-уровня;
  - стандартные фреймворки Security.framework, ExternalAccessory.Framework, Foundation.framework.

Примерный порядок действий пользователя при взаимодействии с приложением

Примерный порядок действий пользователя при первом запуске приложения поддерживающего установку защищённого канала с Токеном

1) пользователь запускает приложение

1.1.1) приложение сообщает пользователю что необходимо включить Bluetooth на устройстве (в случае, если радиоканал Bluetooth на устройстве был включён заранее, пункты 1.1.1-1.1.3 будут пропущены)

1.1.2) приложение показыват кнопку включения Bluetooth (iOS7) или перебрасывает на экран настроек Bluetooth

1.1.3) в случае iOS6 пользователь возвращается в приложение

1.2.1) приложение сообщает пользователю что ему необходимо подключить Токен к устройству (показывает как надо нажать и подержать кнопку на Токене, пока он не заморгает)

1.2.2) Пользователь нажимает кнопку "подключить" и приложение его перебрасывает в настройки Bluetooth где он видит Токен в списке Bluetooth устройств в состоянии "без пары".

1.2.3) Пользователь нажимает на название устройства и ждёт пока произойдёт спаривание

1.2.4) Пользователь возвращается в приложение

2) Приложение находит токен, но при перечислении считывателей посредством вызова **CryptGetProvParam(PP_ENUMREADERS)** по параметру szMedia="NO_MEDIA" возвращенной структуры 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) определяет, что защищенный канал не инициализирован. Приложение сообщает пользователю, что Токен найден, но необходимо будет провести процедуру инициализации и ввести пароль с карточки.

3) Приложение получает пароль от пользователя и через последовательность вызовов функций pkcs11 в итоге получает CKR_OK от функции C_OpenSession

4) Приложение сообщает пользователю что его Токен теперь может работать с этим приложением

5) Приложение работает с контейнером на Токене

Примерный порядок действий при повторном запуске приложения

1) пользователь запускает приложение

1.1.1) приложение сообщает пользователю что необходимо включить Bluetooth на устройстве (в случае, если радиоканал Bluetooth на устройстве был включён заранее, пункты 1.1.1-1.1.3 будут пропущены)

1.1.2) приложение показыват кнопку включения Bluetooth (iOS7) или перебрасывает на экран настроек Bluetooth

1.1.3) в случае iOS6 пользователь возвращается в приложение

1.2.1) приложение сообщает пользователю что ему необходимо подключить Токен к устройству (показывает как надо нажать и подержать кнопку на Токене, пока он не заморгает синим светодиодом)

1.2.2) токен автоматически подключается к устройству

2) Приложение находит токен и при перечислении считывателей посредством вызова CryptGetProvParam(PP_ENUMREADERS) по параметру szMedia="rutoken_ecp_XXXXXXXX" определяет, что инициализация защищенного канала не требуется

3) Приложение работает с контейнером на Токене