

オープンソースのミドルウェア OpenSC

ROBOC 伊藤大輔

OpenSC History

- 2001-11 OpenSC 0.2
 - by Yrjölä and Timo Teräs.
- 2004-12? Belpic
 - Belgium has chosen OpenSC as software basis for their national ID card and the source code changes to OpenSC have been published under LGPL license. Their modified version is published under the name "Belpic", and both complete source code and binaries are available for download. Thanks to Belgium.
- 2006-3 LinuxTag



LinuxTAG 2006



<http://www.opensc-project.org/history.html#LinuxTag2006>

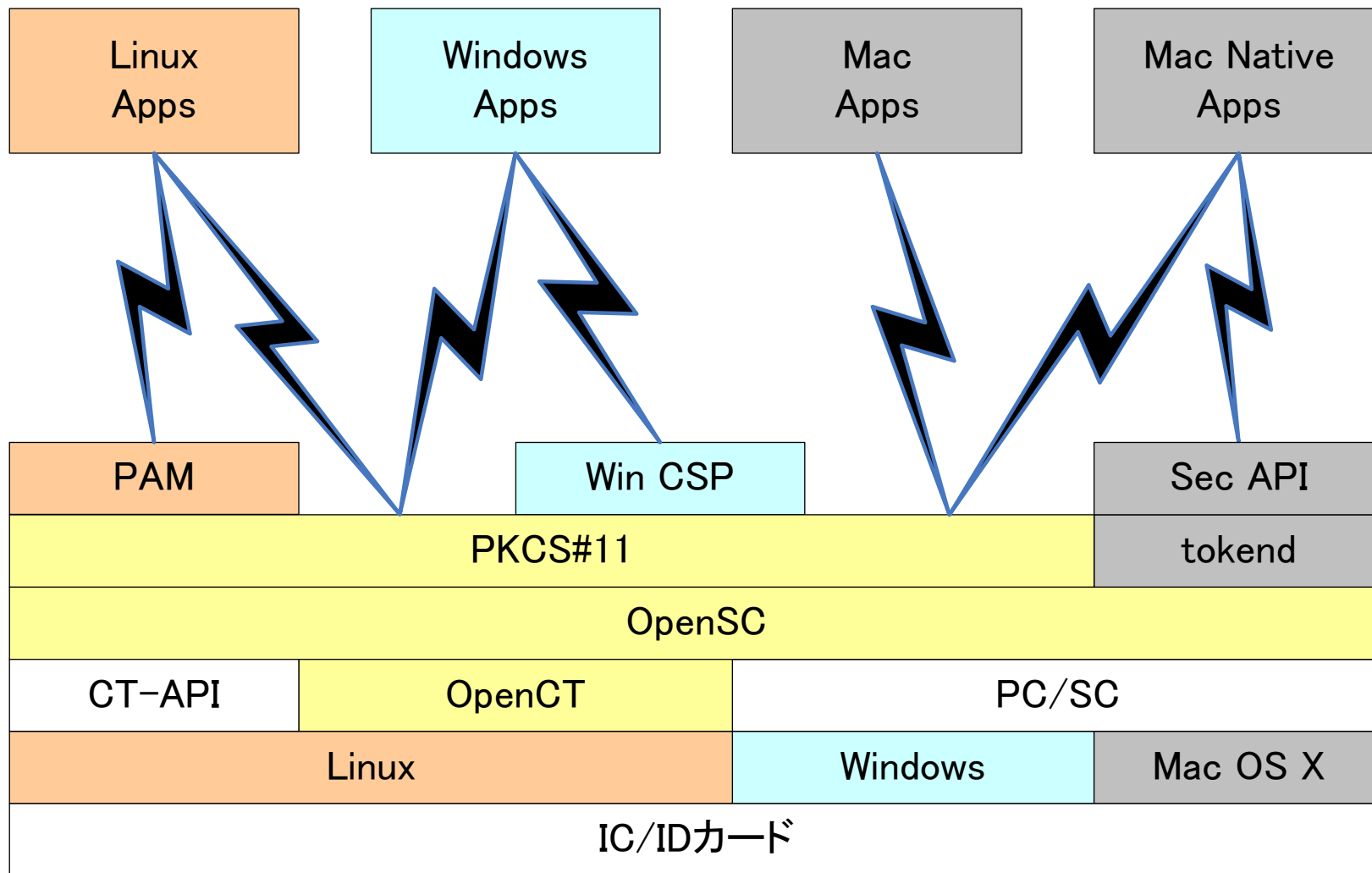
Copyright © 2007 ROBOC Co., Ltd. All rights reserved.



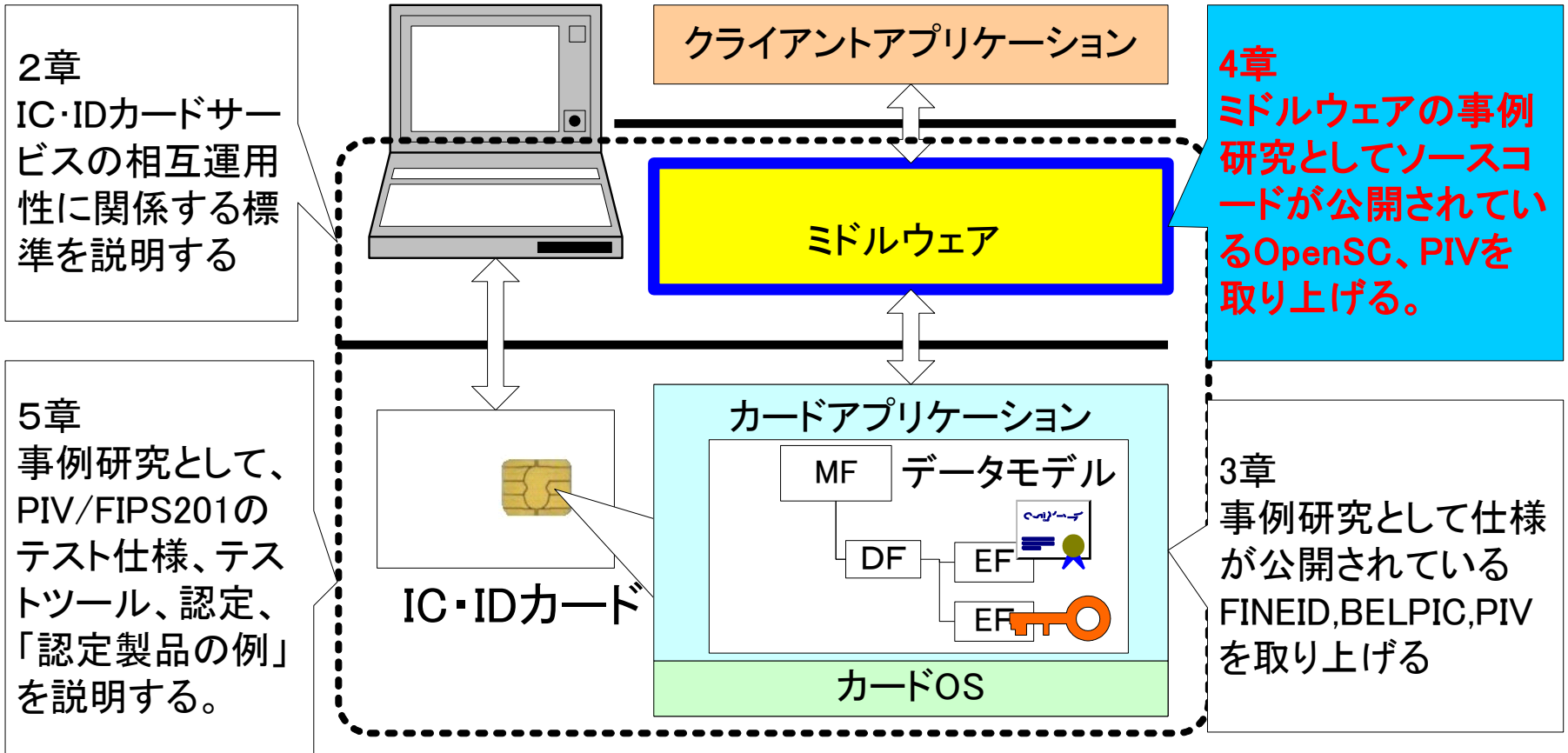
サブプロジェクト

サブプロジェクト名	概要	開発規模
OpenSC	メインプロジェクト。Linux,Mac OS X,Windows上で稼動する、汎用的なカード用ライブラリ、PKCS#11ライブラリ、ツールなど	9万ステップ
Windows Installer SCB	Windows用バイナリパッケージと、Windows独自のツールをサポートする	各種ツールをコンパイルし、まとめたもの。
Apple Mac OS X Installer SCA	Mac OS X用バイナリパッケージと、Macネイティブアプリケーションサポート用プログラム	各種ツールをコンパイルし、まとめたもの。
OpenSSL PKCS#11 Engine	OpenSSL用PKCS#11プログラム。OpenSSLでIC・IDカードを利用できる	1千ステップ
GTK Card	IC・IDカード操作のGUIアプリケーション	8千ステップ
OpenCT	カードリーダー、USBトークン用のドライバ	3万ステップ
Pam PKCS#11	高機能PAM認証モジュール	1万3千ステップ
Pam P11	単機能PAM認証モジュール	1千ステップ
Libp11	PKCS#11ライブラリ	6千ステップ
OpenSC-Java	Java1.5用のIC・IDカードサポート	1万1千ステップ

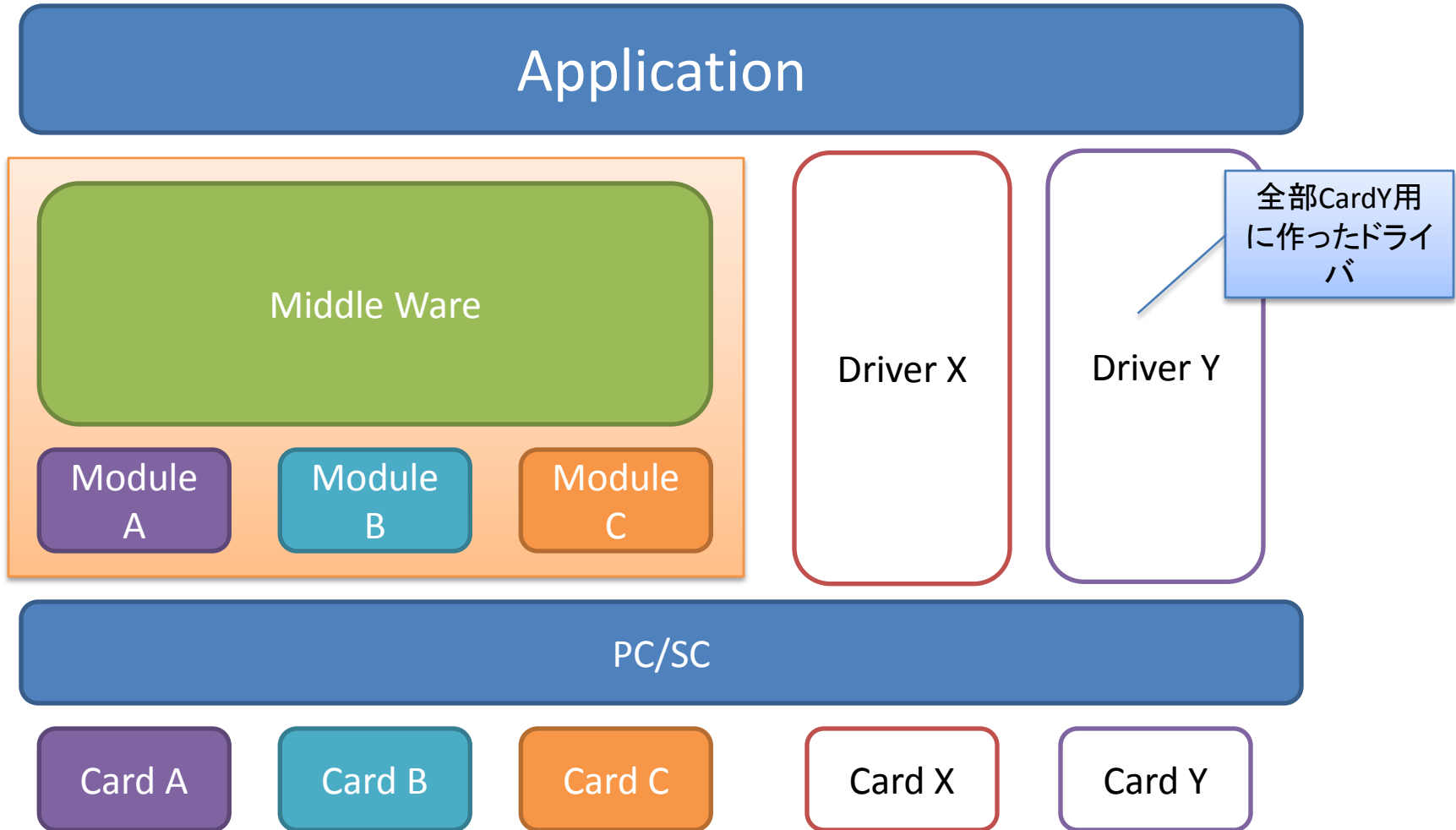
マルチプラットフォーム対応



ミドルウェア



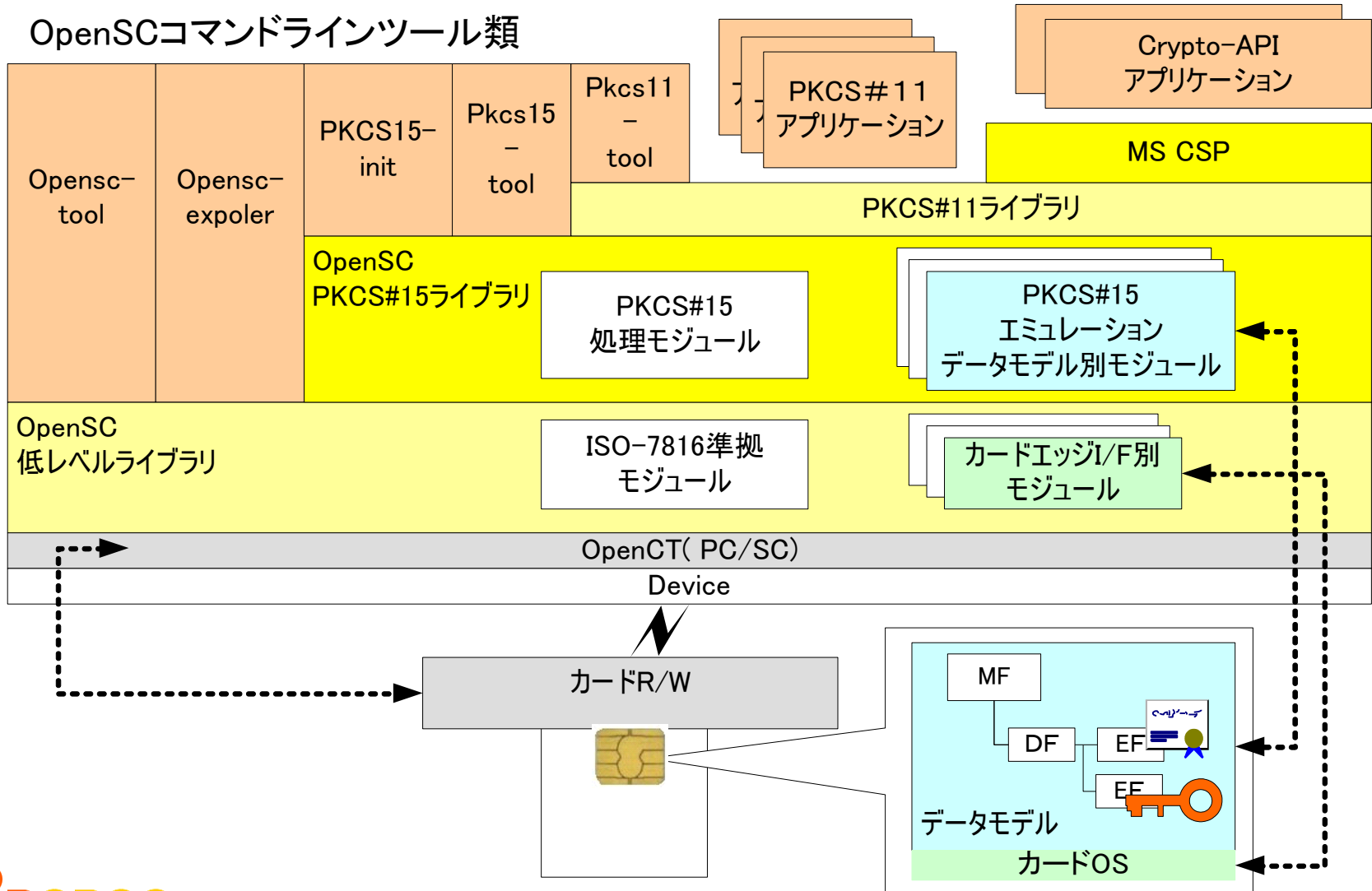
Middle or NOT



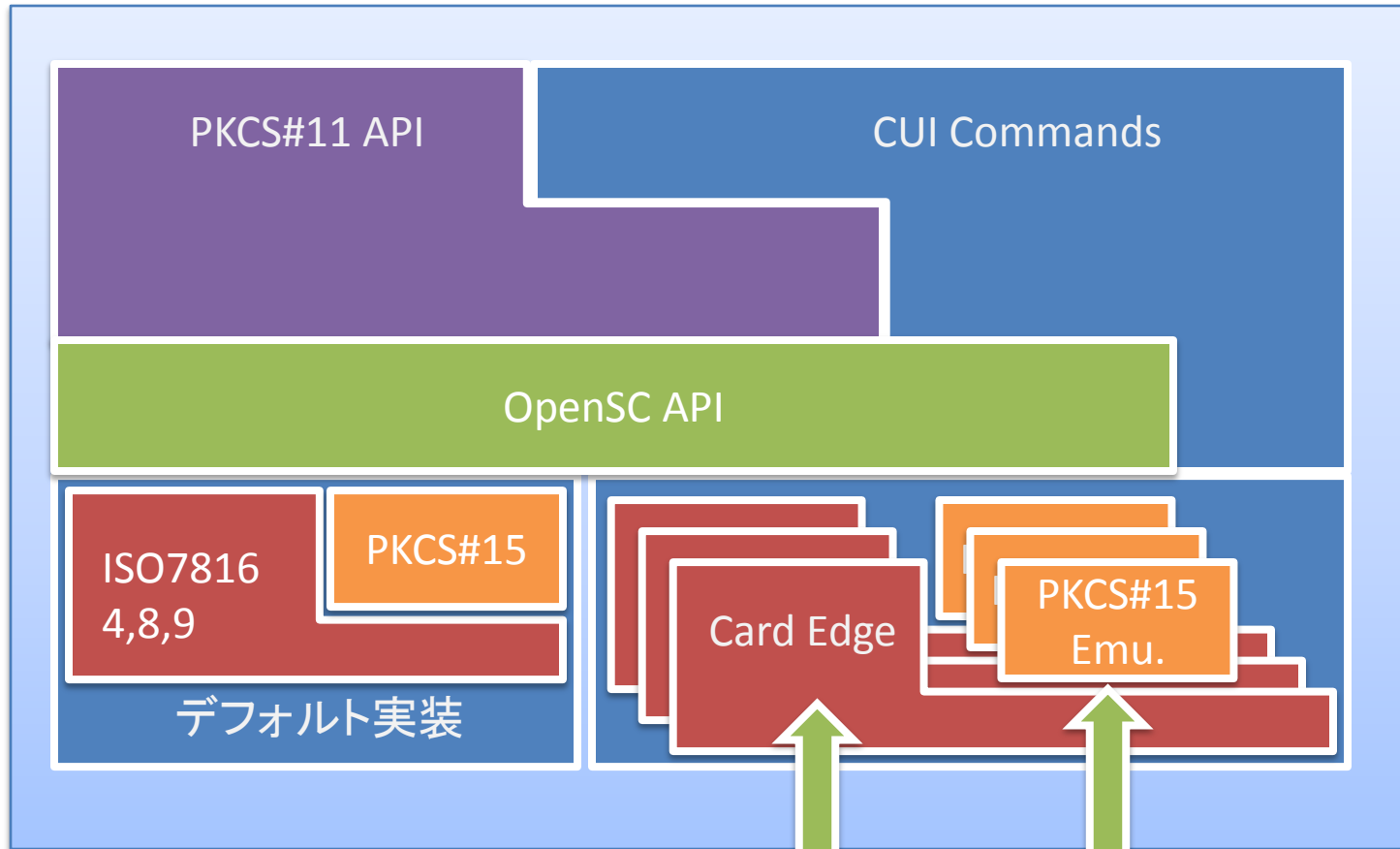
全体のアーキテクチャ

クライアントアプリケーション群

OpenSCコマンドラインツール類



ミドルウェア (OpenSC)



カードエッジI/Fモジュール

PKCS#15エミュレーション
データモデル別モジュール

カード対応モジュール

- カードエッジI/Fモジュール
 - カードエッジI/Fの違いを吸収する
 - カードのOSの違い
 - card-xxx.c
 - デフォルト実装はiso7816.cに書かれている
- PKCS#15エミュレーションモジュール
 - クレデンシャルフォーマットの違いを吸収する
 - カードに格納されるデータの違い
 - pkcs15-xxx.c
 - デフォルト実装はpkcs15.cに書かれている

OpenSCがサポートするカードOS

カード名	説明	
カードOS(カードエッジI/F)		
	Schlumberger/Axalto Cryptoflex	card-flex.c(カードエッジI/F)
	Gemplus GPK	card-gpk.c(カードエッジI/F)
	EMV	card-emv.c(カードエッジI/F)
	Siemens CardOS M4	card-cardos.c(カードエッジI/F)
	IBM JCOP	card-jcop.c(カードエッジI/F)
	Micardo	card-mcrd.c(カードエッジI/F)
	Oberthur	card-oberthur.c(カードエッジI/F)
	OpenPGP	card-openpgp.c(カードエッジI/F)
	Setec Setcos	card-setcos.c(カードエッジI/F)
	Giesecke & Devrient Starcos	card-starcos.c(カードエッジI/F)
	Giesecke & Devrient Seccos	
	TCOS based cards (NetKey E4, SignTrust, Smartkey)	card-tcos.c(カードエッジI/F)

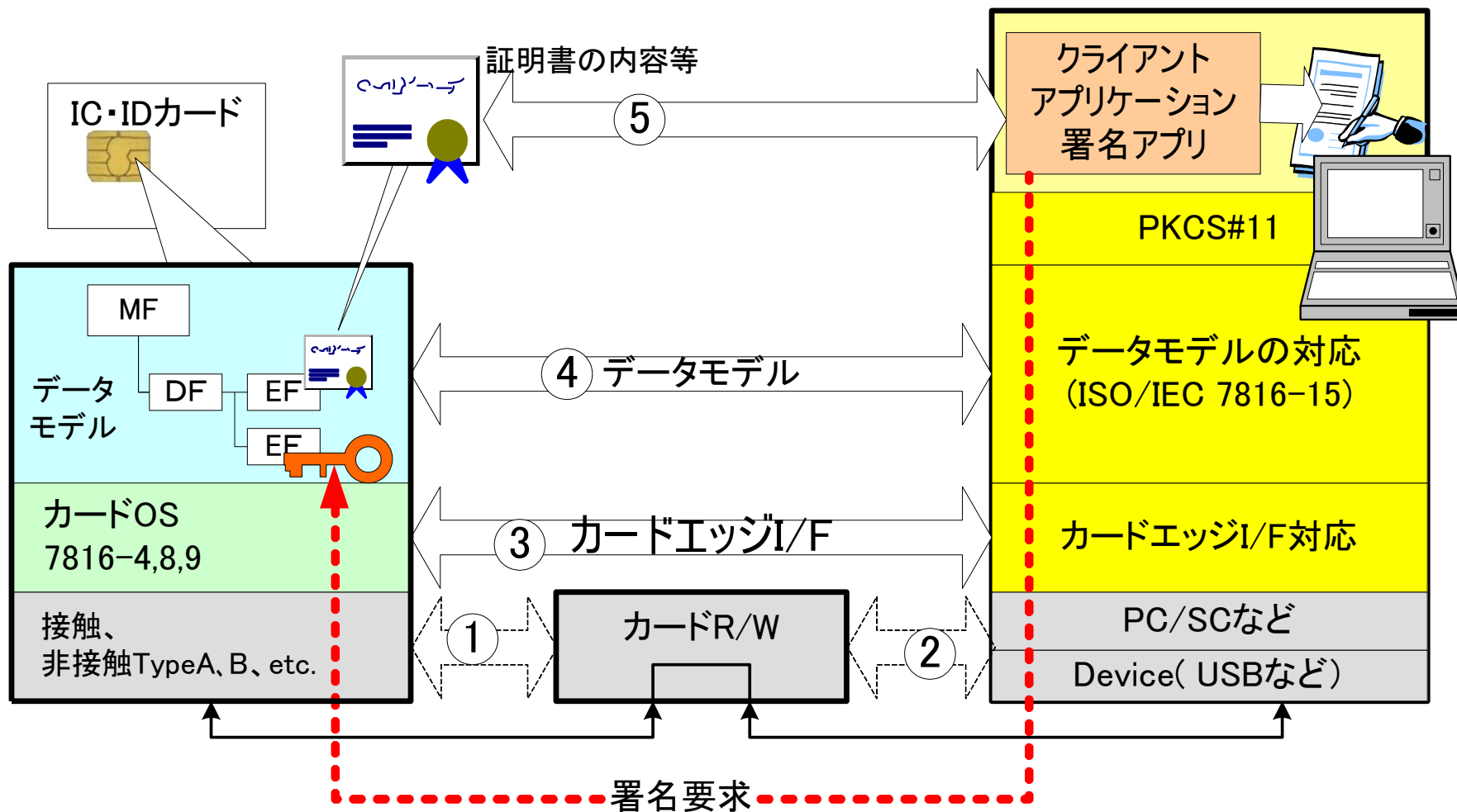
OpenSCがサポートするUSBトークン

カード名		説明
USBトークン		
	Aladdin eToken Pro	card-cardos.c Siemens CardOS M4
	Eutron CryptoIdentity ITSEC	card-starcos.c G&D社のStarcos
	Schlumberger/Axalto e-gate	card-flex.c Schlumberger/Axalto Cyberflex
	Rainbow iKey 3000	card-starcos.c G&D社のStarcos

OpenSCがサポートするIDカード

カード名	説明
国等が発行しているIDカード	
フィンランド ID Card FINEID	PKCS#15準拠カード 。カードOSは、Setec Setcos
スウェーデン Posten eID	PKCS#15準拠カード 。スウェーデンの郵便局のeID
エストニア ID Card EstEID	pkcs15-esteid.c(データモデル) 100万枚以上のIDカードを発行
イタリア Infocamere	pkcs15-infocamere.c(データモデル) イタリアの民間認証局のカード
イタリア Postecert	pkcs15-postecert.c(データモデル) イタリア郵政局の事業会社であるPostecom社が発行
ベルギー-BELPIC	PKCS#15準拠カード 。2006年10月現在400万枚以上のカードを発行。ミドルウェアにOpenSCを全面的に採用
スペイン Ceres	PKCS#15準拠カード 。スペインの電子IDカード
ドイツ ID Cards, eHBA, eGK	pkcs15-tcos.c(データモデル)。ドイツポストの認証局(SignTrust)が発行するカード。ドイツテレコム(Telesec)が発行するカード
台湾	PKCS#15準拠カード 。台湾市民ICカード
オーストリア Bürgerkarte, e-card	pkcs15-atrust-acos.c(データモデル)。オーストリアの市民カード
米国PIV card applet	pkcs15-postecert.c(データモデル)。card-piv.c(カードエッジI/F)

レイヤ



カードエッジインタフェース(1)

ISO/IEC7816-4

コマンドフォーマット

CLA	INS	P1	P2	Lc	コマンドデータ	Le
-----	-----	----	----	----	---------	----

応答フォーマット

応答データ	SW1	SW2
-------	-----	-----

コマンド	CLA	クラスバイト。各ビットに意味を持つ。最上位ビットがセットされている場合、7816標準のコマンドではないことを示す。
	INS	処理を行うコマンドを示す。
	P1	コマンドに付随するパラメータを示す。
	P2	
	Lc	後に続くコマンドデータの長さを示す。
	コマンドデータ	
Le	応答データで予想される最大のデータ長。	
応答	応答データ	
	SW1	コマンド実行結果のステータス（成功・エラー）を示す。
	SW2	

カードエッジインタフェース(2) ISO/IEC7816-4,8

SELECT
MANAGE CHANNEL
READ BINARY
WRITE BINARY
UPDATE BINARY
SEARCH BINARY
ERASE BINARY
READ RECORD
WRITE RECORD
UPDATE RECORD
APPEND RECORD
SEARCH RECORD
ERASE RECORD
GET DATA
PUT DATA
INTERNAL AUTHENTICATE
GET CHALLENGE
EXTERNAL AUTHENTICATE
GENERAL AUTHENTICATE
VERIFY
CHANGE REFERENCE DATA
ENABLE VERIFICATION REQUIREMENT
DISABLE VERIFICATION REQUIREMENT
RESET RETRY COUNTER

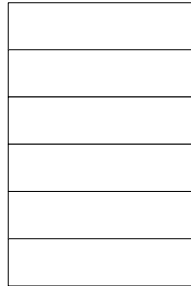
GET RESPONSE
ENVELOPE
MANAGE SECURITY ENVIRONMENT
SET
STORE
RESTORE
ERASE
PERFORM SECURITY OPERATION
COMPUTE CRYPTOGRAPHIC CHECKSUM
COMPUTE DIGITAL SIGNATURE
HASH
VERIFY CRYPTOGRAPHIC CHECKSUM
VERIFY DIGITAL SIGNATURE
VERIFY CERTIFICATE
ENCIPHER
DECIPHER
GENERATE ASYMMETRIC KEY PAIR

実装あり
実装ないが、他のコードから使用
構造体の定義のみアリ

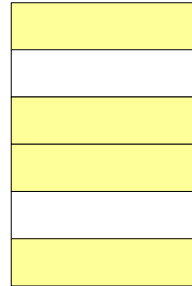
カードエッジI/Fモジュール

```
static struct sc_card_operations iso_ops = {
    no_match,
    NULL,          /* init */
    NULL,          /* finish */
    iso7816_read_binary,
    iso7816_write_binary,
    iso7816_update_binary,
    NULL,          /* erase_binary */
    iso7816_read_record,
    iso7816_write_record,
    iso7816_append_record,
    iso7816_update_record,
    iso7816_select_file,
    iso7816_get_response,
    iso7816_get_challenge,
    NULL,          /* verify */
    iso7816_logout,
    iso7816_restore_security_env,
    iso7816_set_security_env,
    iso7816_decipher,
    iso7816_compute_signature,
    NULL,          /* change_reference_data */
    NULL,          /* reset_retry_counter */
    iso7816_create_file,
    iso7816_delete_file,
    NULL,          /* list_files */
    iso7816_check_sw,
    NULL,          /* card_ctl */
    iso7816_process_fci,
    iso7816_construct_fci,
    iso7816_pin_cmd,
    NULL,          /* get_data */
    NULL,          /* put_data */
    NULL          /* delete_record */
};
```

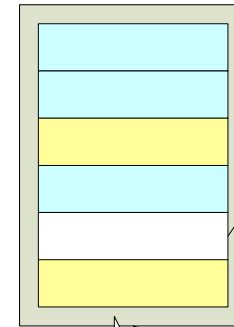
空のオペレーション構造体



標準実装が関数を導入

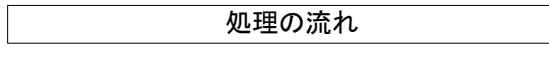


カードエッジI/Fモジュールが上書き



最後まで実装されない関数もある

処理の流れ



最終的にこのオペレーションセットが使用される

```
static struct sc_card_driver *sc_get_driver(void)
{
    if (iso_ops == NULL)
        iso_ops = sc_get_iso7816_driver()->ops;

    belpic_ops.match_card = belpic_match_card;
    belpic_ops.init = belpic_init;
    belpic_ops.finish = belpic_finish;

    belpic_ops.select_file = belpic_select_file;
    belpic_ops.read_binary = belpic_read_binary;
    belpic_ops.pin_cmd = belpic_pin_cmd;
    belpic_ops.set_security_env = belpic_set_security_env;
    belpic_ops.logout = belpic_logout;

    belpic_ops.compute_signature = belpic_compute_signature;
    belpic_ops.get_challenge = iso_ops->get_challenge;
    belpic_ops.get_response = iso_ops->get_response;
    belpic_ops.check_sw = iso_ops->check_sw;

    return &belpic_drv;
}
```

カードエッジI/Fモジュールの選択

- ATR (Answer To Reset)
 - リセットの直後にカードが送ってくるデータ

```
static struct sc_atr_table belpic_atrs[] = {
/* Applet V1.1 */
"3B:98:13:40:0A:A5:03:01:01:01:AD:13:11"
/* Applet V1.0 with new EMV-compatible ATR */
"3B:98:94:40:0A:A5:03:01:01:01:AD:13:10"
/* Applet beta 5 + V1.0 */
"3B:98:94:40:FF:A5:03:01:01:01:AD:13:10 "
```
- これを見て、使用するモジュールを選択する

カード中のデータを読み出す

- 例

- iso7816_select_file(“3F005015” ...);

- ファイルを選択して...

- iso7816_read_binary(...);

- 読み出す

データモデル別モジュール

- OpenSCが**必要とするデータ**をカードから読みだす
- デフォルト実装
 - カードに格納されているデータがPKCS#15準拠であることが前提
 - PKCS#15準拠カードから、OpenSCが**必要とするデータ**を集めるコード
- IDカード個別実装: PKCS#15エミュレーションカードモジュール
 - PKCS#15非準拠カードから、OpenSCが**必要とするデータ**を集めるコード
 - PKCS#15準拠カードでは**不要**

PKCS#15

- カードに、どんなデータが入っているのか？という設計図自身が、カードに入っている
- 設計図は決められた場所に保存されている
- データは、「これはどんなデータだ」という属性とともに、保存される
 - 例：読み出し禁止フラグ
 - 「ONだと読み出せない」ではなく、「読み出せないものはONになっている」
 - 指示ではなく、表明

PKCS#15

ICカード

MF(3F00)

EF.DIR(2F00)
どんなアプリケーションがあるか

ほかのアプリケーション

DF.CIA アプリケーション(例: 5015)

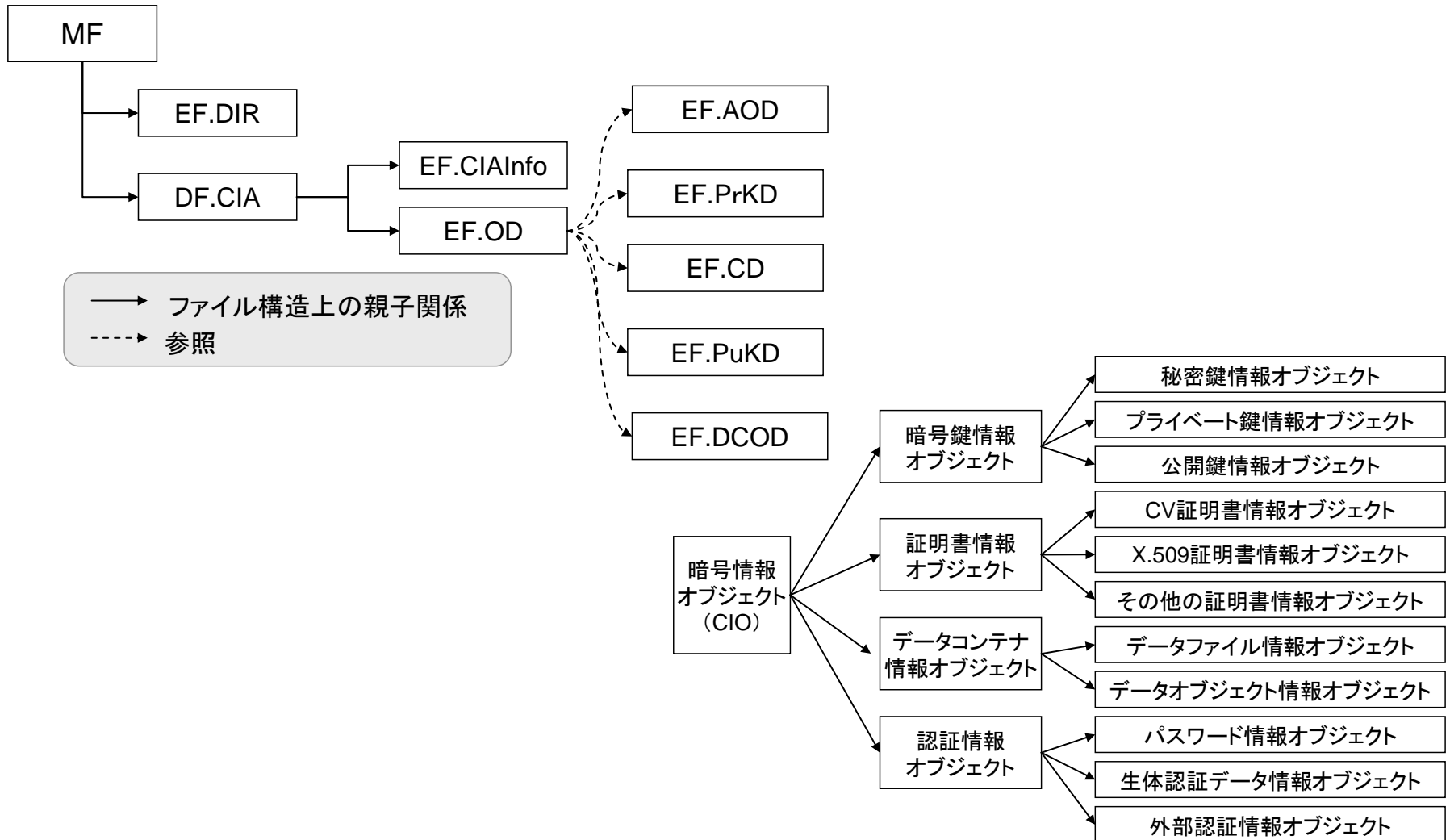
EF.OD(5031)
何が、どこにあるか?

EF.PrKD データの属性
実際のプライベート鍵

EF.AOD データの属性
実際のPINコード

EF.CIO

PKCS#15



BELPICモジュール

- カードエッジI/Fモジュール
 - card-belpic.c
 - 1587ステップ
 - function
 - select_file
 - read_binary
 - pin_cmd
 - set_securyt_env
 - compute_signature
 - decipherは空のまま
 - BELPICはdecipherをサポートしていない。
- PKCS#15エミュレーションモジュール
 - BELPICはPKCS#15準拠なので、なし

```
static struct sc_card_driver belpic_drv = {  
    "Belpic cards",  
    "belpic",  
    &belpic_ops,  
    NULL, 0, NULL  
};
```

```
static struct sc_card_driver *sc_get_driver(void)  
{  
    if (iso_ops == NULL)  
        iso_ops = sc_get_iso7816_driver()->ops;  
  
    belpic_ops.match_card = belpic_match_card;  
    belpic_ops.init = belpic_init;  
    belpic_ops.finish = belpic_finish;  
  
    belpic_ops.select_file = belpic_select_file;  
    belpic_ops.read_binary = belpic_read_binary;  
    belpic_ops.pin_cmd = belpic_pin_cmd;  
    belpic_ops.set_security_env = belpic_set_security_env;  
    belpic_ops.logout = belpic_logout;  
  
    belpic_ops.compute_signature = belpic_compute_signature;  
    belpic_ops.get_challenge = iso_ops->get_challenge;  
    belpic_ops.get_response = iso_ops->get_response;  
    belpic_ops.check_sw = iso_ops->check_sw;  
  
    return &belpic_drv;  
}
```

belpic_compute_signature

```
static int belpic_compute_signature(sc_card_t *card, const u8 * data,
    size_t data_len, u8 * out, size_t outlen){
    int r;
    r = iso_ops->compute_signature(card, data, data_len, out, outlen);
#ifdef HAVE_GUI
    if (r == SC_ERROR_SECURITY_STATUS_NOT_SATISFIED && SSO_OK(card->ctx)) {
        r = belpic_askpin_verify(card, SCR_USAGE_AUTH);
        if (r == 0)
            r = iso_ops->compute_signature(card, data, data_len, out, outlen);
    }
#endif
    return r;
}
```

iso7816_compute_singature

```
static int iso7816_compute_signature(sc_card_t *card,
    const u8 * data, size_t datalen,
    u8 * out, size_t outlen)
{
    int r;
    sc_apdu_t apdu;
    u8 rbuf[SC_MAX_APDU_BUFFER_SIZE];
    u8 sbuf[SC_MAX_APDU_BUFFER_SIZE];

    assert(card != NULL && data != NULL && out != NULL);
    if (datalen > 255)
        SC_FUNC_RETURN(card->ctx, 4, SC_ERROR_INVALID_ARGUMENTS);

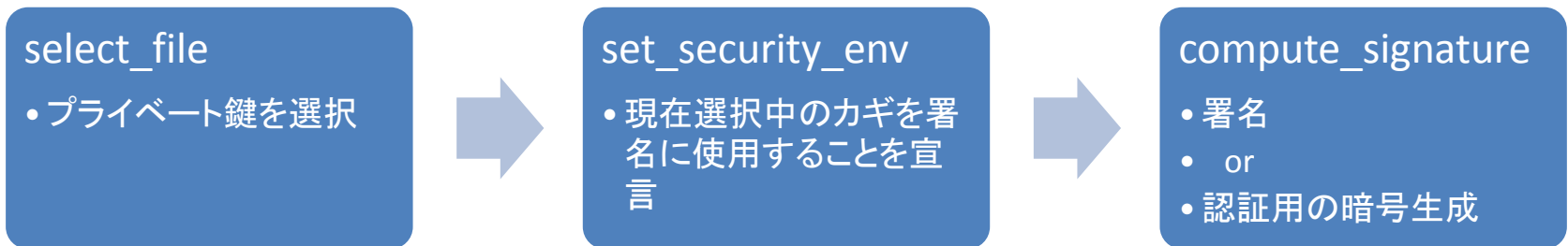
    /* INS: 0x2A PERFORM SECURITY OPERATION
     * P1: 0x9E Resp: Digital Signature
     * P2: 0x9A Cmd: Input for Digital Signature */
    sc_format_apdu(card, &apdu, SC_APDU_CASE_4_SHORT, 0x2A, 0x9E, 0x9A);
```

PERFORM SECURITY
OPERATION
CARD COMMAND

だけしか呼んでいない！

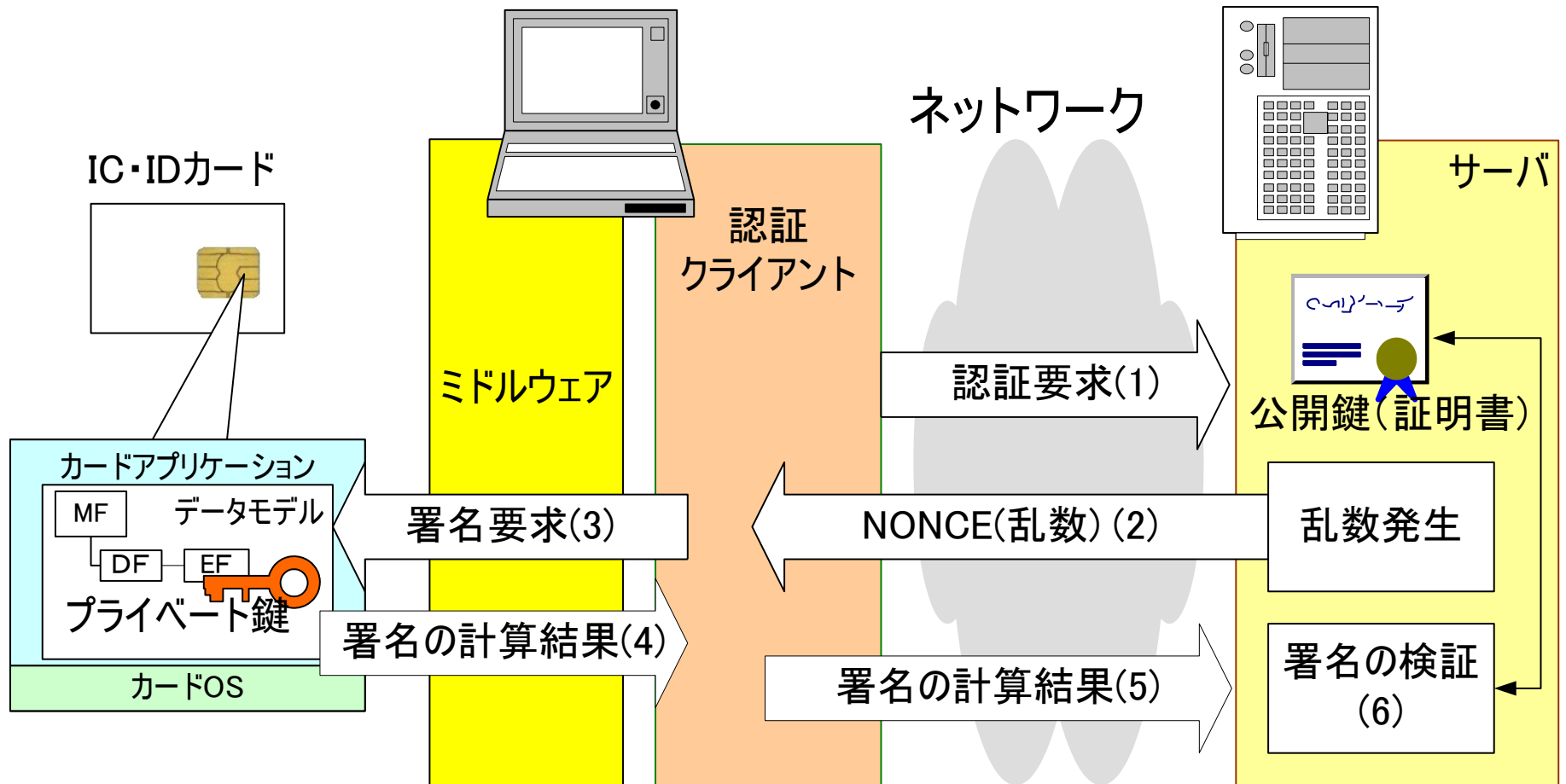
- ダイジェストアルゴリズム？
- プライベート鍵？
- 認証？

署名/認証



- カードコマンド一発では完了しない
- COMPUTE SIGNATUREコマンドは、選択されているプライベート鍵の「使用目的」によって意味が異なる
 - 否認防止用のカギなら、署名
 - 認証用のカギなら、認証
 - 与えられたデータをプライベート鍵で暗号化して送り返す. 公開鍵で複合できれば、認証OK！！

COMPUTE SIGNATUREで認証？



set_security_env(belpic)

```
apdu.le = 0;
apdu.data = sbuf;
apdu.resplen = 0;

r = sc_transmit_apdu(card, &apdu);
SC_TEST_RET(card->ctx, r, "Set Security Env APDU transmit failed");

r = sc_check_sw(card, apdu.sw1, apdu.sw2);
SC_TEST_RET(card->ctx, r, "Card's Set Security Env command returned error");
```

```
if (*env->key_ref == BELPIC_KEY_REF_NONREP) {
#ifdef HAVE_GUI
    r = belpic_askpin_verify(card, SCR_USAGE_SIGN);
    if (r != 0 && r != SC_ERROR_KEYPAD_CANCELLED)
        sc_error(card->ctx, "Verify PIN in SET command returned %d¥n", r);
    else
        sc_debug(card->ctx, "Verify PIN in SET command returned %d¥n", r);
#else
    sc_debug(card->ctx, "No GUI for NonRep key present, signature cancelled¥n");
    return SC_ERROR_NOT_SUPPORTED;
#endif
}
```

否認防止用のカギだったら、毎回PINの入力が必須

PKCS#11とカードエッジI/F 署名 (Axalto Cryptoflex 32Kの例)

PKCS#11の関数	カードエッジI/Fの呼び出し回数
C_LoadModule	
C_Initialize	SELECT * 10回 READ BINARY *21回
C_GetSloatList	
C_OpenSession	
C_Login	SELECT * 1回 VERIFY * 1回
C_Sign	SELECT FILE * 1回 (署名生成コマンド * 1回)
C_CloseSession	(終了コマンド*1回)
C_Finalize	

署名生成コマンドと終了コマンドは非標準I/F

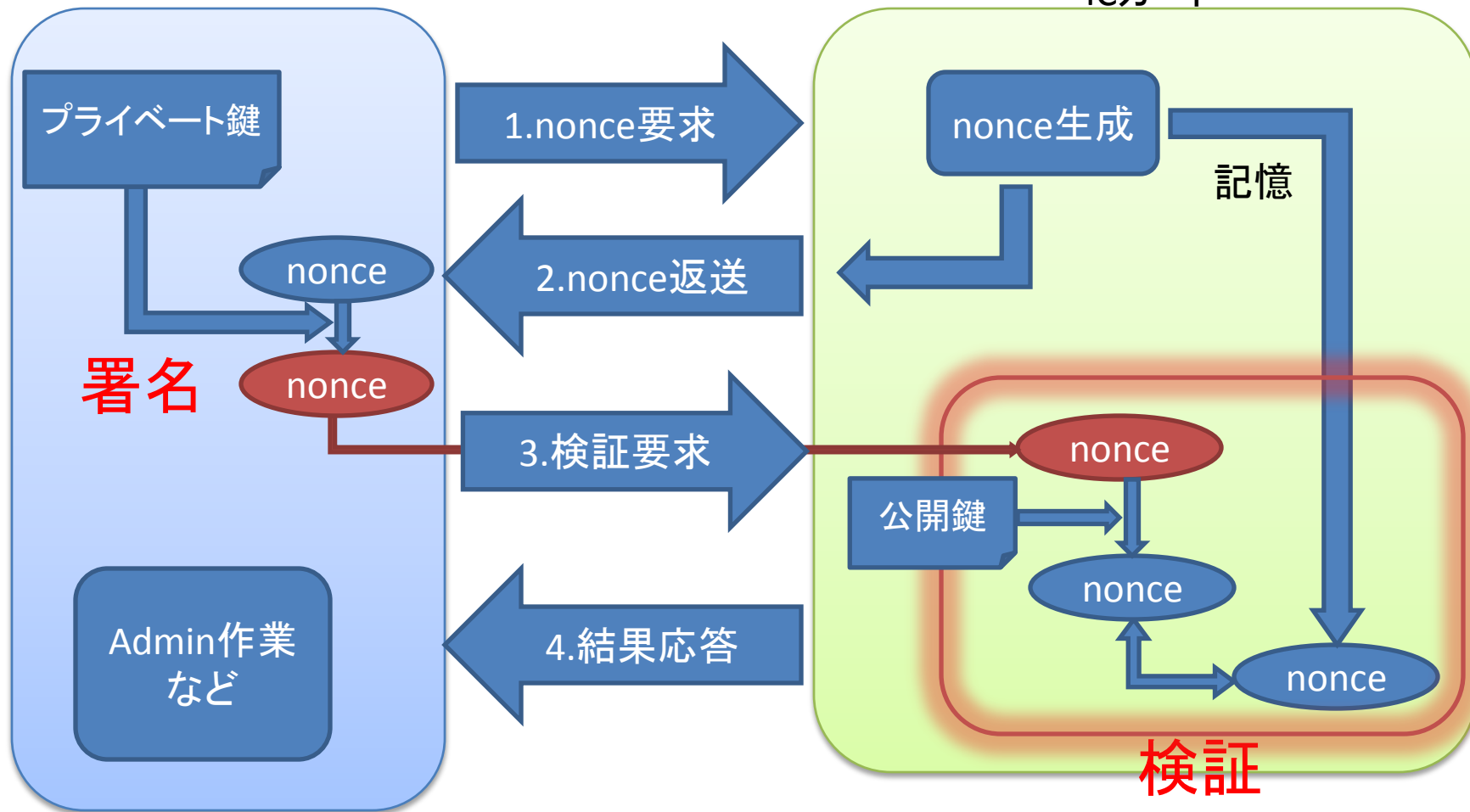
カードエッジI/Fを知れば。。。。 すべてわかるつもりだったんだけど...(;^_^A

- 認証
 - カードホルダを認証する
 - VERIFY
 - 外部認証
 - EXTERNAL AUTHENTICATE
 - GENERAL AUTHENTICATE
 - 内部認証
 - INTERNAL AUTHENTICATE
- 署名
 - PERFORM SECURITY OPERATION(Compute Signature)
- 暗号化
 - PERFORM SECURITY OPERATION(Encipher)
- 復号
 - PERFORM SECURITY OPERATION(Decipher)

外部認証: 正規なソフトウェア or 管理者? (PIVの例)

ICカード管理アプリケーション

ICカード



アプリケーション側の責任が結構たくさんある

- 鍵の管理

- 例えば外部認証・相互認証に使う鍵が安全に管理されなければならない。ICカードを使うと、鍵がカードから外に出ないから安全、というのはICカードを使ったサービスのごく一部しか見ていない。

- 正しい手順

- 単純にカードエッジI/Fを呼び出すだけでは操作が完了しない。

- バグのないコード

- ICカードが高度に安全でも、それを使うアプリケーションにバグがあれば、全然だめ。

オープンなミドルウェアがあれば

- 難しい仕様を、みんなで実装できる
 - 思い込みや勘違いを防げる
 - 「秘密だから安全」ではない解決策
- ソースを見て、仕組みを理解できる
 - 仕様のみから仕組みを理解するのは難しい
 - ソースから仕様を理解するのも難しいが...

参考

- OpenSC
 - <http://www.opensc.org>
- BELPIC Java Applet
 - <http://lists.musclecard.com/pipermail/muscle/2005-September/004387.html>
- IC・ID カードの相互運用可能性の向上に係る基礎調査
 - <http://www.ipa.go.jp/security/fy18/reports/ICID/index.html>
 - シーズ編
 - http://www.ipa.go.jp/security/fy18/reports/ICID/seeds_rep.pdf