



Central department of security and information systems

Security Architecture

Date of Creation	: October 22, 2007
Project name	: TRUECRYPT
Reference	: SPM030-ARC
Version	: 2.00
Classification	: public
Status	: Issued
Number of Pages	: 30 (including 2 headers)
Comments	: none

序文

特に教育の現場での、本書の複製、および/または二次利用は、次の3つの条件を満たすことによって、国防総局情報セキュリティ中央局(SGDN/DCSSI)によって許可されている。

- 本書の配布を無償とすること。

- 本書を複製する場合には、完全性に注意を払うこと(原書に忠実であること): 修正、改ざんは認めない。

- 本書の複製版には、例えば「本書は、国防総局情報セキュリティ中央局(SGDN/DCSSI) (<http://www.ssi.gouv.fr/en>)によって策定されたものである。」など、出目を明らかにする文言を含めること。

目次

1. 序章	6
1.1. コモン・クライテリアについて	6
1.2. 参照文献	6
1.3. TOE モジュールの種類について	7
2. TSF が維持するセキュリティ・ドメイン	8
2.1. メモリ内の機密性の高いデータ保護	8
2.2. スレッド管理	8
2.3. 暗号コンテキスト	8
3. TOE の初期化におけるセキュリティ保護	9
3.1. 「TrueCrypt.exe」の初期化	9
3.1.1. 説明	9
3.1.2. 「TrueCrypt.exe」の起動プロセス	10
3.2. 「TRUECRYPT.FORMAT.EXE」の初期化	11
3.2.1. 説明	11
3.3. 「TRUECRYPT.SYS」ドライバの初期化	14
3.3.3. 説明	14
4. 自己保護	16
4.1. 物理的な自己保護手段	16
4.2. 論理的な自己保護手段	16
4.2.1. 完全性制御メカニズム	16
4.2.2. 暗号鍵(マスタ鍵)の保護	18
4.2.3. カーネル・モードの実行	18
5. SFR 実施メカニズムのバイパス防止	21
5.1. 乱数の生成	21
5.2. パスワード処理	22
5.3. PKCS#5 V2.0 による(鍵の)導出	22
5.4. 暗号アルゴリズム	22
5.5. ハッシュ関数	23
5.6. 秘密情報の削除	24
付録 A 略語と頭字語	25
A.1. 略語と頭字語	25
A.2. 定義	25
付録 B 参照文献	26
付録 C コモン・クライテリア関連の情報	27
C.1. 序章	27
C.2. ADV_ARC.1 コンポーネントの目的	27
C.3. メソッドロジ	28

C.3.1. ドメイン分離.....	28
C.3.2. 初期化プロセスにおけるセキュリティの維持.....	28
C.3.3. 自己保護.....	28
C3.4. バイパス防止.....	28
C.3.5. ADV_ARC.1 コンポーネントの要件.....	28
C.3.6. CEM の基本的なタスク.....	29



図 1. TOE のモジュール表現.....	7
図 2. TrueCrypt.exe の開始プロセス.....	10
図 3. TrueCrypt.Format.exe の開始プロセス.....	12
図 4. TrueCrypt.sys ドライバの初期化プロセス.....	15
図 5. ヘッダの復号プロセス.....	17
図 6. 暗号鍵の保護.....	18
図 7. 保護リング.....	18
図 8. カーネル・モードの実行.....	20

用語

CC:	コモン・クライテリア
CEM:	情報技術セキュリティ評価のための共通方法
SFR:	セキュリティ機能要件
SFR 実施:	最低でも 1 つの SFR の要素を実装、または、SFR を実施する要素を直接サポートする。
SFR 支援:	SFR 実施の構成要素にも依存するが、SFR を実装するが直接的な役割はない。
SFR 非干渉:	SFR の実装において果たす役割がない。
TOE:	評価対象。評価対象とは、TOE は評価の対象となる製品、またはシステムの一部である。
TSF:	TOE セキュリティ機能
TSFI:	TSF インタフェース

1. 序章

本書ではデータ・ディスクの暗号化ソフトウェア・アプリケーション製品である TrueCrypt のセキュリティ・アーキテクチャを記述する。

本書には、コモン・クライテリアの ADV_ARC.1 コンポーネントの要件に対応する情報を含む。

1.1. コモン・クライテリアについて

このファミリ(ADV_ARC)の目的は、開発者による TSF のセキュリティ・アーキテクチャの記述の提供である。これにより別のコンポーネントにおいて立証された分析結果と共に、TSF が期待通りの特性を有していることを確認できる。TOE のセキュリティ・アーキテクチャによって、TSF を調査すれば TOE のセキュリティ評価を正しく実施することができる。妥当なアーキテクチャなしでは TOE の機能全体を検査しなければならない。

1.2. 参照文献

本書では、基本的に以下をベースに構成されている。

- セキュリティ・ターゲット[ST]、
- 機能仕様[ADV_FSP]、
- TOE 設計[ADV_TDS]、および
- TrueCrypt 暗号仕様[CRYPTO_SPEC]。

1.3. TOE モジュールの種類について

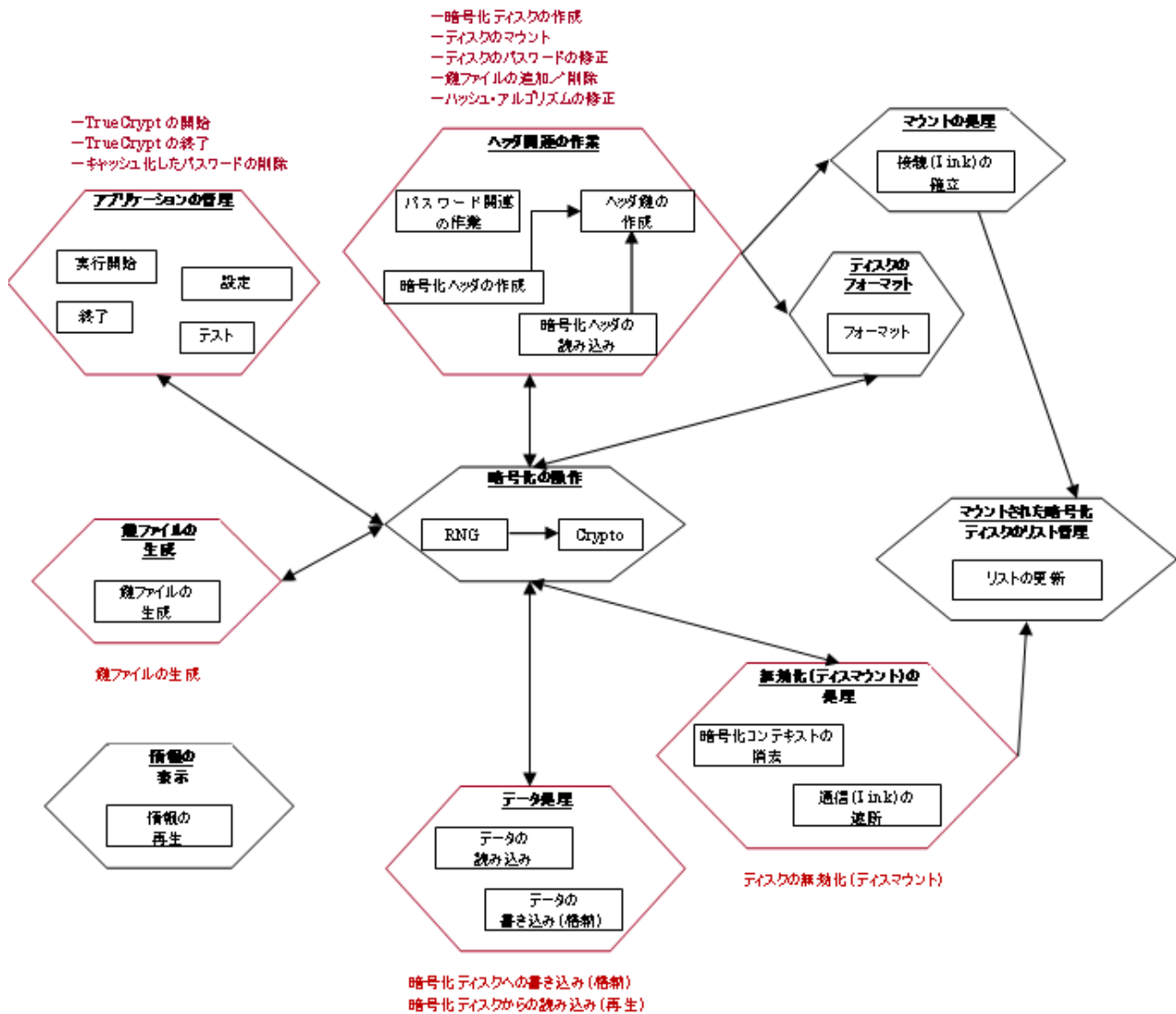


図 1: TOE のモジュール表現

- TOE の 1 つ以上の TSFI と接続されているサブ・システム
- Tsfi** : サブ・システムと接続されている TSFI のリスト

TSFI とサブ・システム間のトレーサビリティに関する表は、[ADV_TDS]文書の第 5 章を参照のこと。

2. TSF が維持するセキュリティ・ドメイン

アプリケーションには、次のような 3 つのタイプのセキュリティ・ドメインが存在する。

- メモリ内の機密性の高いデータ保護
- スレッド管理
- 暗号化ディスク毎の暗号コンテキストの生成

2.1. メモリ内の機密性の高いデータ保護

TrueCrypt は初期化される時、*VirtualLock* 関数を使用して機密性の高いデータ(パスワードやマスタ鍵など)を格納するメモリをロックするようオペレーティング・システムに要求する。

この関数は、第 3.1 章「TrueCrypt.exe」の初期化で説明する。

ドライバでもデータをロックすることはできるが、この場合はカーネル内であり Windows の *ExAllocatePoolWithTag* を使用する。

2.2. スレッド管理

ディスクがマウントされると、ディスクに関するさまざまな要求(読み出し、書き込み、アンマウント)を管理するために、カーネル・スレッド(カーネルはスレッドの状態を直接管理することができる)が生成される。

スレッドが生成されると、関数の引数は(Windows の *ExAllocatePoolWithTag* 関数により)メモリにロックされ、SWAP ファイルにスワップアウトされないようになる。ロックされる引数は DeviceObject (TrueCrypt のマウント済みボリュームに対応した Windows ストラクチャ)へのポインタ、ボリュームのアクセス・パス、ドライバ番号などがあるが、とりわけ「マウント」ストラクチャには、マウント用データ(例えば修正済のパスワードなど)が含まれる。

ユーザの要求はまず TrueCrypt ドライバへ送られ、ドライバによりユーザの要求に対応するディスクのスレッドへ転送される(22 ページの図 8:カーネル・モードの実行を参照)。受信側のスレッドはユーザの要求をキューに格納し、順番に対応する。

このスレッドはカーネルの一部であるため、使用されるメモリは共通である。このメモリはオペレーティング・システムにより管理され、メモリをスレッドごとに割り当てている。

スレッドでは、次のような管理が可能である。

- 様々なディスクに対する同期要求より生じる問題の回避。
- ディスク間のデータの組織化、構造化(マスタ鍵)。

2.3. 暗号コンテキスト

TrueCrypt は、暗号化ディスクごとに暗号コンテキストを作成する。

コンテキストとは、TrueCrypt により動的に割り当てられ RAM にロックされる構造体である (*crypto_open* 関数は、Windows の *ExAllocatePoolWithTag* 関数によりロックされる)。このコンテキストは、ディスクを暗号化するために使用するマスタ鍵とアルゴリズムにより特徴付けられる。コンテキストの一意性は、乱数より生成されたマスタ鍵により保証される。

暗号化ディスクは、対応する暗号コンテキストが使用された場合に限り読み出すことができる。

3. TOE の初期化におけるセキュリティ保護

本章では、TrueCrypt を構成する 2 つ実行ファイルの初期化を把握することが重要である。

- 「TrueCrypt.exe」実行ファイル： 初期化時のセキュリティ保護を実施するのは「Start」モジュールで、これは「アプリケーション管理」サブ・システムに属する
- 「TrueCrypt.Format.exe」実行ファイル： 初期化時のセキュリティ保護を実施するのは「Formatting」モジュールで、これは「ディスクフォーマット」サブ・システムに属する

ドライブの初期化については、第 3.3 章で詳しく説明している。

注:

TrueCrypt は誰でも実行することができるが、管理者権限を持つユーザがドライバをロードしなければならない。仮に、未許可のユーザがドライバのロードなしに TrueCrypt を実行しようとする、アプリケーションは終了してしまう。

3.1. 「TrueCrypt.exe」の初期化

3.1.1. 説明

アプリケーションは、TrueCrypt ドライバをロードし開始される(ドライバが既にロードされている場合はアタッチする)。障害時には、アプリケーションは正しく初期化できず終了する。

ドライバがロード(または、アタッチ)されると、機密性の高いグローバル変数は TrueCrypt によりメモリにロックされるため、SWAP ファイルへスワップアウトされることはない。

最終的に、このアプリケーションは自身のインタフェースを実行し、既にマウントされている(すなわち、TrueCrypt 開始前にドライバが既にメモリにある場合)ボリュームをリストする。

注:

1. 「TrueCrypt Format.exe」と異なり、乱数生成器は「TrueCrypt.exe」起動時にのみ初期化される。乱数生成器は、例えば鍵ファイルを作成する場合など、必要な場合に限って使用される。
2. 実行ファイルのソース・コードは、プロセスが終了すると初期化時に使用したコンポーネントへアクセスすることができないように作成されている。

3.1.2. 「TrueCrypt.exe」の起動プロセス

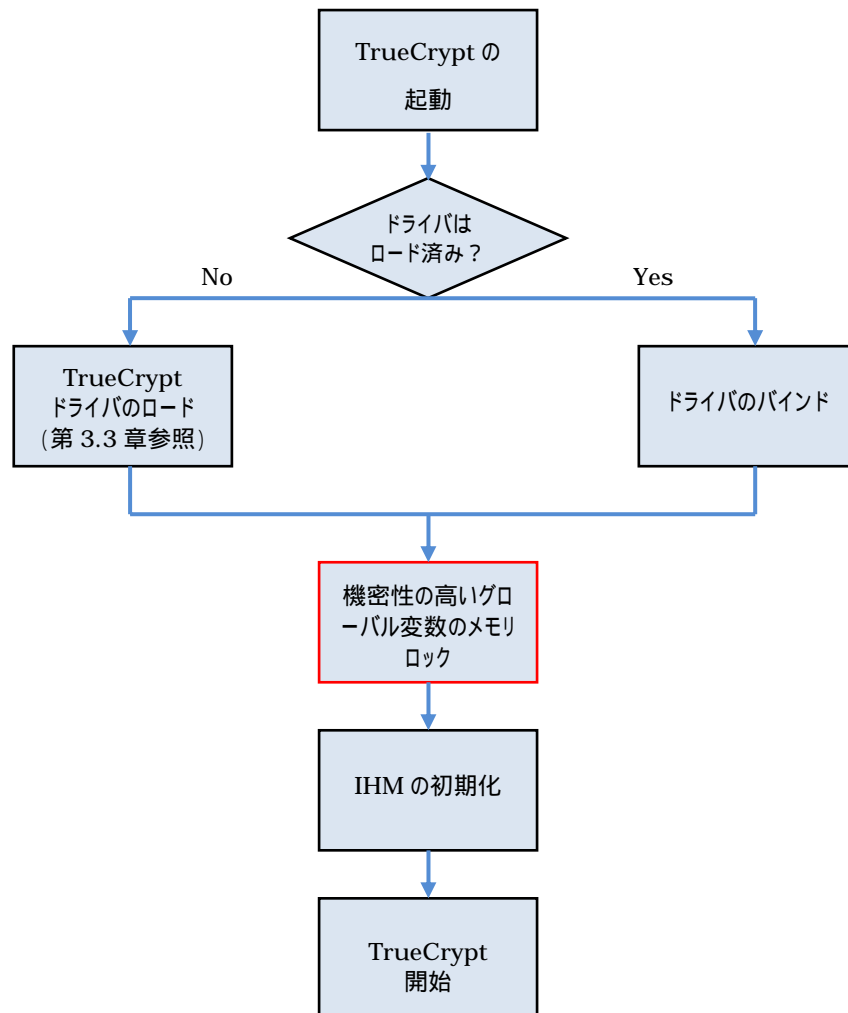


図 2: TrueCrypt.exe の開始プロセス

機密性の高いグローバル変数のメモリロック Windows の *VirtualLock* 関数を使用することによりメモリに機密性の高いグローバル変数(ボリュームパスワード、マウントオプションなど)をロックし SWAP ファイルにスワップアウトされないようにしている。

*VirtualLock*関数は、ページフォルトが発生しないようにプロセスのある仮想アドレススペースの領域を物理メモリにロックする。

VirtualLock 関数の説明:

```
BOOL VirtualLock (  
LPVOID lpAddress,  
SIZE_T dwSize  
);
```

lpAddress: ロックするページ領域のベース・アドレス用ポインタ

dwSize: ロックする領域のサイズ(バイト)。このページ領域は、lpAddress パラメタから (lpAddress + dwSize) の範囲に、1 つ以上のバイトを含む全てのページが該当する。2 バイトが、あるページにまたがっている場合には、双方のページがロックされる。

以下は、このロックを実行している TrueCrypt のコードである。

```
VirtualLock (& VolumePassword, sizeof (VolumePassword));  
VirtualLock (& CmdVolumePassword, sizeof (CmdVolumePassword));  
VirtualLock (& MountOptions, sizeof (MountOptions));  
VirtualLock (& defaultMountOptions, sizeof (defaultMountOptions));
```

3.2. 「TRUECRYPT.FORMAT.EXE」の初期化

3.2.1. 説明

「TrueCrypt Format.exe」は、ボリューム生成を補助するプログラムである。このプログラムは個別のプログラムとして直接実行することもできるが、「TrueCrypt.exe」メインプログラムより実行することもできる。

最初に「TrueCrypt Format.exe」が、乱数生成器を初期化する。この機能は、暗号化ディスクを作成するために必要である。

次に「TrueCrypt.exe」と同様に、ドライバをロードするか自身にアタッチし使用できるようにする。

一旦ロード(アタッチ)されると、アプリケーションは標準的なデータを使用し、さまざまなアルゴリズムをテストする。

最終的に HMI が初期化され、ボリューム生成の補助ができるようになる。

注:

実行ファイルのソース・コードは、プロセスが終了すると初期化時に使用したコンポーネントへアクセスすることができないように作成されている。

上記の補助が可能になると、「暗号化オプション」ページから「暗号アルゴリズムテスト」コンポーネントを手動で開始することができる。ただしこのインタフェースはアプリケーションを変更することはできない。

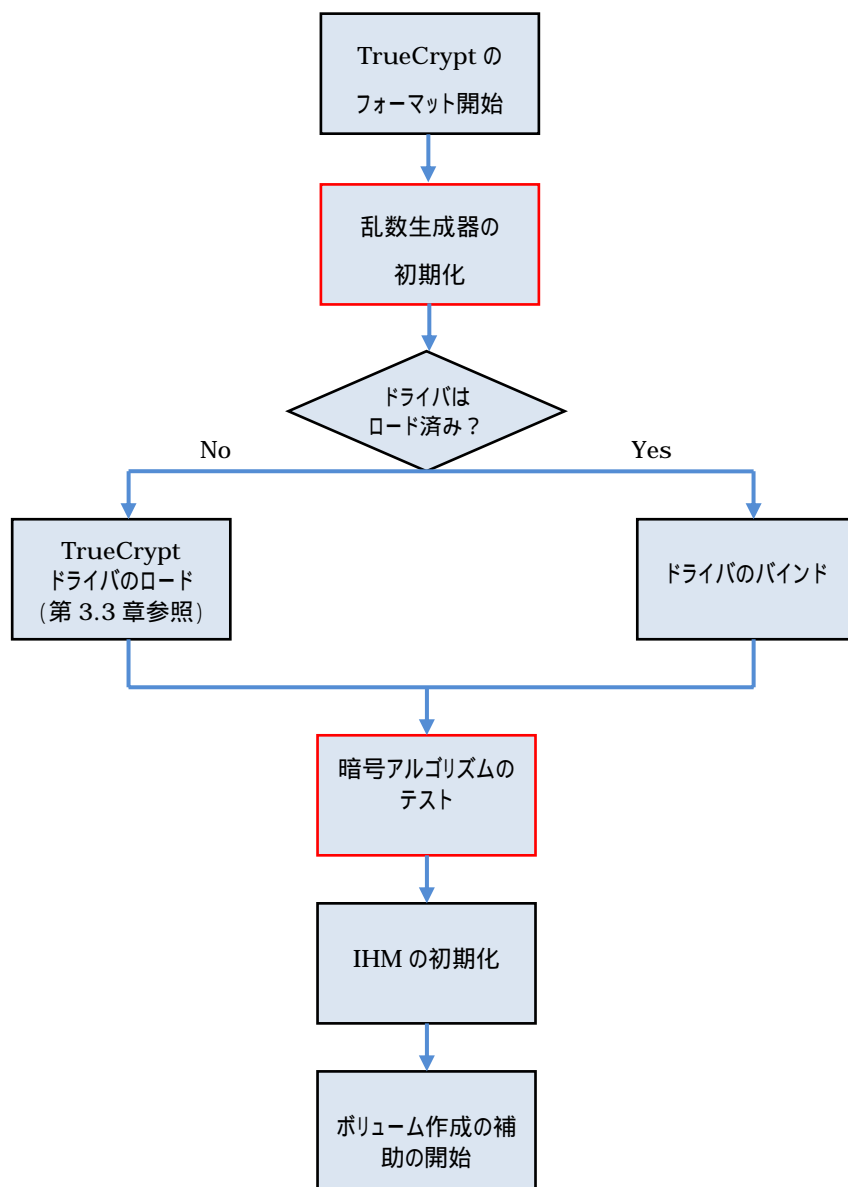


図 3: TrueCrypt.Format.exe の開始プロセス

乱数生成器の初期化 初期化は Randinit 関数により実行される。この関数は、メモリを乱数プールに割り当てロックする。また Randinit 関数は、乱数プール提供スレッドを生成し乱数プールを提供するためにキーボードとマウスをフックする。

暗号アルゴリズムテスト TrueCrypt は、利用可能なアルゴリズムをテストすることができるように標準入出力用データを備えている。テストが失敗するとアプリケーションは終了する。

Randinit 関数

```
Int Randinit ()
{
    if (bRandDidInt) return 0;

#ifdef WIN32
    InitializeCriticalSection (&critRandProt);
#endif
    bRandDidInt = TRUE;

    pRandPool = (unsigned char *) TAlloc (RANDOMPOOL_ALLOCSIZE);
    if (pRandPool == NULL)
        goto error;
    else
        memset (pRandPool, 0, RANDOMPOOL_ALLOCSIZE);

#ifdef WIN32
    VirtualLock (pRandPool, RANDOMPOOL_ALLOCSIZE);

    hKeyboard = SetWindowsHookEx (WH_KEYBOARD, (HOOKPROC)&KeyboardProc,
    NULL, GetCurrentThreadId ());
    if (hKeyboard == 0) handle Win32Error (0);

    hMouse = SetWindowsHookEx (WH_MOUSE, (HOOKPROC)&MouseProc, NULL,
    GetCurrentThreadId ());
    if (hMouse == 0)
    {
        handle Win32Error (0);
        goto error;
    }
    if (!CryptAcquireContext (&hCryptProv, NULL, NULL, PROV_RSA_FULL, 0)
        && !CryptAcquireContext (&hCryptProv, NULL, NULL, PROV_RSA_FULL,
    CRYPT_NEWKEYSET))
    {
        hCryptProv = 0;
    }
    if (_beginthread (ThreadSafeThreadFunction, 8192, NULL) == -1)
        goto error
#endif
    return 0

error:
    Randfree ();
    Return 1;
}
```

AES アルゴリズムをテストする TrueCrypt コード(抜粋):

```
/* AES*/

for (i = 0; i < AES_TEST_COUNT; i++)
{
    init cipher = AES;
    memcpy (key, aes_ecb_vectors[i].key, 32);
    memcpy (tmp, aes_ecb_vectors[i].plaintext, 16);
    CipherInit(cipher, key, ks_tmp);

    EncipherBlock(cipher, tmp, ks_tmp);
    if (memcmp(aes_ecb_vectors[i].ciphertext, tmp, 16)!= 0)
        break;

    DecipherBlock(cipher, tmp, ks_tmp);
    if (memcmp(aes_ecb_vectors[i].plaintext, tmp, 16)! = 0)
        break;
}
if (i != AES_TEST_COUNT)
    bFailed = TRUE;
```

3.3. 「TRUECRYPT.SYS」ドライバの初期化

3.3.1. 説明

ドライバの役割は、システムと TrueCrypt 間の中継である。

ロードされると、ドライバはパラメタを初期化しさまざまな暗号アルゴリズムをテストする。

その後、ドライバがロードされ使用可能になる。

注:

実行ファイルのソース・コードは、プロセスが終了すると初期化時に使用したコンポーネントへアクセスすることができないように作成されている。

メインプログラムがロードされると、「暗号化アルゴリズムテスト」コンポーネントを手動で開始することができる。ただしこのインタフェースはアプリケーションを変更することはできない。

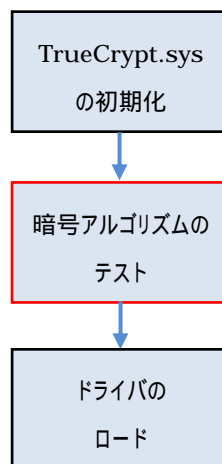


図 4: TrueCrypt.sys ドライバの初期化プロセス

暗号アルゴリズムのテスト ドライバの初期化で実行されるテストシーケンスは、TrueCrypt.Format.exe 開始時と同じである。

TrueCrypt は、さまざまなアルゴリズムをテストできるように、標準入出力用のデータを備えている。TrueCrypt は、アルゴリズムを使用してこれらの標準データの暗号化 / 復号を実行し、その結果を標準データと比較照合する。

ただしドライバの場合でテストが失敗した場合、明示的なエラー表示はないが、ユーザはボリュームをマウントすることができない。

4. 自己保護

4.1. 物理的な自己保護手段

TrueCrypt には、物理的な自己保護は実装されていない。

4.2. 論理的な自己保護手段

4.2.1. 完全性制御メカニズム

通常の暗号化ディスクのヘッダ			
位置(バイト)	サイズ(バイト)	状態	説明
0	64	未暗号化	シード(暗号化ディスクの作成時に、無作為に生成される。)
64	4	暗号化	ASCII 文字列「TRUE」(復号されたヘッダが有効かを検証するために使用される完全性パターン)
68	2	暗号化	ヘッダのバージョン番号
70	2	暗号化	暗号化ディスクのマウントに必要な最小 TrueCrypt バージョン
72	4	暗号化	(復号されたヘッダが有効な場合)検証に使用される(復号された)CRC32 チェックサム最後の 256 ビット
76	8	暗号化	暗号化ディスク作成日
84	8	暗号化	ヘッダ最終修正日
92	8	暗号化	予約(ゼロ)
100	156	暗号化	未使用
256	可変	暗号化	二次マスタ鍵(LRW モード)
288	可変	暗号化	マスタ鍵

復号されたヘッダの完全性の検証には、2 組のブロック(1 組 4 バイト)があれば可能である。

- 最初の条件である復号されたヘッダが有効であることを確認する完全性のパターン、
- 2 番目の条件である復号されたヘッダが有効であることを確認する復号されたマスタ鍵の CRC32 チェックサム。

これらにより、復号されたマスタ鍵が有効であり、ヘッダが改ざんされていないことを保証する。

このチェックが不完全な場合、ヘッダは復号されない。

以下は、ヘッダの復号プロセスである。

4.2.2. 暗号鍵(マスタ鍵)の保護

マスタ鍵は、乱数生成器を使用して生成される。従って、推測されにくい一意的な鍵を生成することができる。これらの鍵は、暗号化ディスクのデータを暗号化 / 復号するために使用される。マスタ鍵のサイズは、ユーザが選択した暗号アルゴリズムによって決定される。

TrueCrypt では、これらの暗号鍵を暗号化ディスクのヘッダに格納する。

これらの暗号鍵を保護するために、このヘッダは、鍵ファイルのリスト、暗号化ディスクのシード、ユーザのパスワードから生成されたヘッダ鍵によって暗号化される。

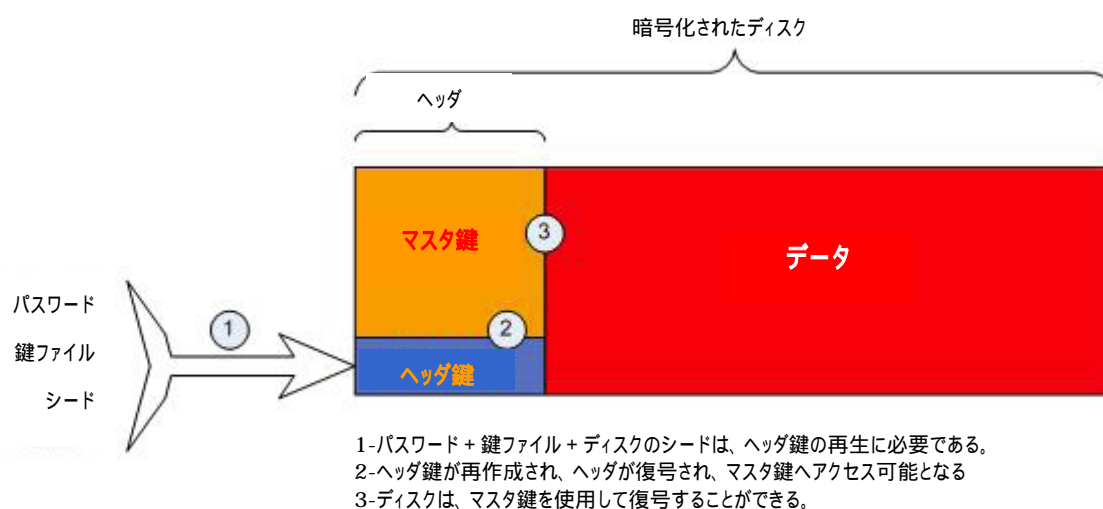


図 6: 暗号鍵の保護

セキュアな暗号アルゴリズム (AES、TripleDES)、および導出アルゴリズム (PKCS#5v2) に基づくこのプロセスにより、マスタ鍵が暗号化ディスクのヘッダに含まれることとなる。

4.2.3. カーネル・モードの実行

例えば、x86 のような最近のプロセッサは、「リング」と呼ばれる権限、すなわち保護システムを実装している。これらのリングは、最上位の権限 (レベル 0) から最低位の権限 (レベル 3) までの 4 つのリングが階層的に配されている。あるレベルから別なレベルへのパスは厳しく制限されており、メモリへのアクセスはレベルによって制限されている。

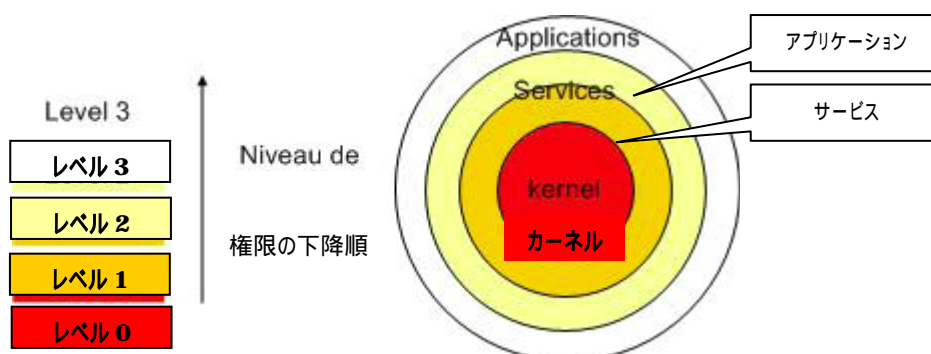


図 7: 保護リング

現在のオペレーティング・システムでは、これら4つのリングのうち、実際に使用されているのは2つだけであることが多い。

- レベル0(カーネル・モードとも呼ばれる) : メモリに全面的にアクセスすることができるマイクロプロセッサの実行コードで、マイクロプロセッサに関連の全ての機能を使用することができる。このコードは、アプリケーションが誤動作した場合でも、システム全体にリスクが及ぶことはないことを保証する。
- レベル3(ユーザ・モードとも呼ばれる) : ユーザ用のソフトウェアが動作するレベル。システムのセキュリティを確保するために全面的に管理される(ある種の命令は実行できない、ある種のメモリ領域にはアクセスできない)。

TrueCrypt は、次のようなオペレーションを実行するために、カーネル・レベルの「Truecrypt.sys」ドライバを使用する。

- ディスクのマウント： ヘッドは、ロックされたメモリ領域で復号される (*readBuffer* 関数は、Windows の *ExAllocatePoolWithTag* 関数でロックされる)
- アンマウント
- キャッシュされたパスワード消去
- 暗号化ディスクからの情報の読み出し
- ディスク上の読み / 書き要求の管理。日付の暗号化、および復号は、ロックされたメモリ領域で実行される (*tmpBuffer* 関数は、Windows の *ExAllocatePoolWithTag* 関数でロックされる。)

ドライバがこれら全ての要求を受け付け適切なスレッドに振り分け、要求されたオペレーションが実行される。

このようにして、カーネル・レベルで実行されるオペレーションのセキュリティを確保することができる。

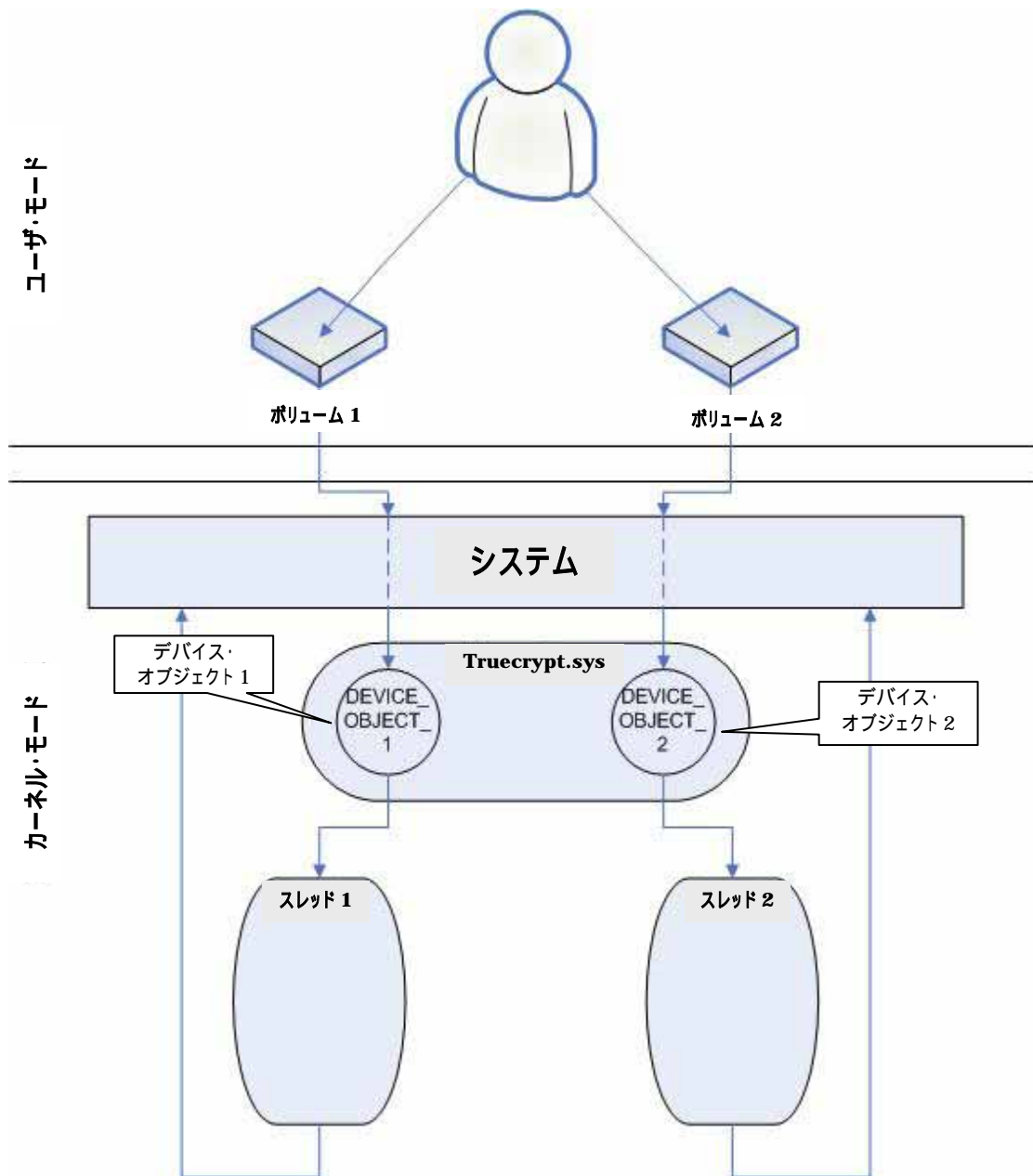


図 8: カーネル・モードの実行

5. SFR 実施メカニズムのバイパス防止

設計文書では、SFR 実施モジュールを次のように定義している。

SFR 実施モジュール
TrueCrypt 停止
パスワード処理
ヘッダ鍵導出
暗号化ヘッダ作成
暗号化ヘッダ読み出し
フォーマット
リンク削除
暗号コンテキスト削除
データ読み出し
データ書き込み
RNG
暗号化オペレーション

これらのモジュールで実装されている SFR 実施メカニズムは以下の通りである。

- 乱数生成
- パスワード処理
- ヘッダ鍵導出
- 暗号化 / 復号メカニズム
- ハッシュ関数
- 秘密情報の削除

注:

ディスクをアンマウントする際に使用するリンク削除のメカニズムは、オペレーティング・システムをベースにしている。従って、TrueCrypt ではこのメカニズムのバイパス防止を保証することはできない。

5.1. 乱数の生成

TrueCrypt の乱数生成器は、一般的な標準仕様ではない。この乱数生成器では、乱数プールに格納されたオペレーティング・システムにより提供される統計的なデータと、キーボード操作、マウスの移動よりの入力データを使用する。

乱数生成器は、マスタ鍵、二次(マスタ)鍵、シードとフォーマット用のランダムデータを生成するために使用される。乱数生成器の設計、および実装方法は、次のような研究がベースになっている。

- Peter Gutmann による Software Generation of Practically Strong Random Numbers
- Carl Ellison による Cryptographic Random Numbers

乱数生成器の機能は、「暗号化仕様」[CRYPTO_SPEC]文書 28 ページの第 3.2.1.章「乱数生成」で説明している。

5.2. パスワード処理

TrueCrypt では、鍵ファイルに含まれるバイナリ・データとユーザのパスワードを組み合わせ、パスワードを変換することができる。

鍵ファイルはさまざまな周辺装置 (ハードディスク、USB スティック、CD-ROM など) に格納することができ、暗号化ディスクをマウントするにはこれらの周辺装置が必須となるため、より高いセキュリティを提供できる。

暗号化に使用するアルゴリズムとその補足的な情報は、「暗号化仕様」[CRYPTO_SPEC]文書 34 ページの第 3.2.1.3 章「ヘッダ鍵生成」のユーザ・パスワード処理にて説明している。

5.3. PKCS#5 V2.0 による (鍵の) 導出

この機能を使用することで、変換された暗号化ディスクのパスワードとシードからヘッダ鍵を導出することができる。

PKCS#5 は RSA ラボラトリーの標準であり、ヘッダ鍵生成の信頼性を保証している。

PKCS#5 の機能は、「暗号化仕様」[CRYPTO_SPEC]文書 36 ページの第 3.2.1.3 章「ヘッダ鍵生成」の PKCS#5v2 に基づく (鍵の) 導出 (Derivation according to PKCS#5v2) で説明している。

5.4. 暗号アルゴリズム

TrueCrypt は、次のような属性を有する 6 通りの暗号アルゴリズムを提供する。

暗号アルゴリズム	ブロック・サイズ(ビット)	鍵サイズ(ビット)	利用可能な暗号モード
AES	128	256	LRW, CBC
Blowfish	64	448	LRW, CBC
CAST5	64	128	LRW, CBC
Serpent	128	256	LRW, CBC
3DES	64	56*3	LRW, CBC
Twofish	128	256	LRW, CBC

AES (Rijndael): 米国の政府機関が使用する新たな暗号化標準として、2000 年に、NIST (米国標準技術局: National Institute of Standards and Technology) によって選択された対称暗号アルゴリズム。

Blowfish: 大きな鍵を生成することができる対称暗号アルゴリズム。このアルゴリズムは 16 ラウンドで構成されており、現時点では完全に信頼できる。このアルゴリズムを破る唯一の手法は総当たり攻撃のみである。150 種以上もの製品 (市販製品、またはフリーソフト) にこのアルゴリズムが使用されている (このアルゴリズムが使用されている製品のリストは、<http://www.schneier.com/blowfish-products.html> を参照のこと)。

CAST5: 政府が使用するために、カナダの通信安全保障局 (CSE: Communication Security Establishment of Canada) が承認する対称暗号アルゴリズム。

Serpent: AES 選択において、(上で紹介した新たな暗号化標準) Rijndael アルゴリズムに

次いで二番目となった対称暗号アルゴリズム。

- TripleDES: DES アルゴリズムを 3 回連続し適用する対称暗号アルゴリズム。3DES は (NIST により) 標準化され、よく知られ実装方法も簡単なアルゴリズムだが、その処理速度が遅いため AES のような最新のアルゴリズムにより代替されてきている。
- Twofish: 完全版においては、このアルゴリズムを対象とする攻撃は存在しない対称暗号アルゴリズム。Blowfish の場合同様、このアルゴリズムを破る唯一の手法は総当たり攻撃である。

世界的に認識されているこれらのアルゴリズム¹を使用することで、データ暗号化への信頼性とデータを解読しようとする攻撃への耐性が保証される。

これらのアルゴリズムには、次のような 2 通りの暗号モードがある。

- LRW: このモードは、暗号化ハードディスクのセクタ用に、特別に研究されたものである。このモードは、SYSWG (セキュリティ・イン・ストレージワーキング・グループ) によって標準化されている。TrueCrypt は、128 ビットのブロック、および 64 ビットのブロックごとにデータを処理する LRW を実装している。
- CBC: ブロックチェイニングによる暗号化。CBC モードは平文のブロックに、排他的論理和を使用し暗号済みのブロックを組み合わせることによる、連鎖構造を持つ暗号である。TrueCrypt の CBC モードは、CBC の標準パターンに従って実装されていない。平文データが暗号化されると、「ホワイトニング」との排他的論理和が実行される。

詳しい情報は、「暗号化仕様」[CRYPTO_SPEC]文書 39 ページの第 3.3.1 章「暗号化 / 復号」、44 ページの第 4.1.1. 章「標準的な暗号化と復号アルゴリズムテスト」、および 48 ページの第 4.1.3 章「標準的な LRW アルゴリズムテスト」を参照のこと。

5.5. ハッシュ関数

TrueCrypt では、乱数の生成、および鍵の導出に使用する 3 つのハッシュ関数²を提供する。

- RIPEND-160: セキュリティが確保されていない 128 ビットのハッシュ関数 (MD4、MD5、RIPEND) の代替として考案される。160 ビットのハッシュ値を生成する。
- SHA-1: 国家安全保障局 (NSA: National Security Agency) が設計し、連邦情報処理規格 (FIPS: Federal Information Processing Standard) として米国政府により公開される。このハッシュ関数は 160 ビットのハッシュ値を生成する。これまでいくつかの攻撃が知られているが、現在最も効率的な攻撃でも、ハッシュ値の衝突を発見するためには、 2^{63} 回の SHA-1 オペレーションを実行する必要があると予測されている。

¹ AES と Twofish のみがこの評価の対象である。

² RIPEND-160、および SHA-1 のみがこの評価の対象である。

WHIRLPOOL: (AES アルゴリズムにも関わった) Vincent Rijmen らによって設計された。このハッシュ関数は 512 ビットのハッシュ値を生成する。このアルゴリズムは、堅牢かつ信頼性を保証するために、修正した AES 暗号化ブロックを使用する。Whirlpool は、乱数の生成に特化した ISO / IEC 10118-3 標準で承認されている。

詳しい情報は、「暗号化仕様」[CRYPTO_SPEC]文書 49 ページの第 4.1.4 章「ハッシュ・アルゴリズムのシードに関するテスト」を参照のこと。

5.6. 秘密情報の削除

TrueCrypt では、アプリケーションの実行に必要ななくなった機密性の高いデータは、直ちに削除することをポリシーにしている。データの削除は、次のような 2 通りの方法で実施される。

- Burn 関数(ptr, size)の使用: ptr は削除すべきメモリ領域のアドレスであり、size はその領域のサイズをバイトで示したものである。この関数は、メモリ領域を 0xff の値で埋め、次に 0 で上書きする。
- Memset 関数(ptr, 0, size)の使用: ptr は削除すべきメモリ領域のアドレスであり、size はその領域のサイズをバイトで示したものである。この関数は、メモリ・ゾーンを 0 で上書きする。

付録 A： 略語と頭字語

A.1. 略語と頭字語

CC: コモン・クライテリア

CEM: 情報技術セキュリティ評価のための共通方法

SFR: セキュリティ機能要件

ST: セキュリティ・ターゲット

セキュリティ・ターゲットには、開発者の視点で捉えた特定の製品のセキュリティ要件が記載される。セキュリティ機能の仕様(要約仕様)を含み、1 つ以上のプロテクション・プロファイル(PP)に定義されたセキュリティ要件を含むこともある。

TOE: 評価対象

TOE は、評価対象となる製品、またはシステムの一部である。

TSF: TOE セキュリティ機能

TSFI: TSF インタフェース

A.2. 定義

スレッド: スレッドとは、コードやそのスタックによって特徴付けられる実行単位である。

付録 B: 参照文献

- [CC1] *Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and general model. Version 3.1, Revision 1, September 2006. CCIMB-2006-09-001.*
- [CC2] *Common Criteria for Information Technology Security Evaluation, Part 2: Security functional requirements. Version 3.1, Revision 1, September 2006. CCIMB-2006-09-002.*
- [CC3] *Common Criteria for Information Technology Security Evaluation, Part 3: Security assurance requirements. Version 3.1, Revision 1, September 2006. CCIMB-2006-09-003.*
- [CEM] *Common Methodology for Information Technology Security Evaluation, Evaluation Methodology. Version 3.1, Revision 1, September 2006. CCIMB-2006-09-004.*
- [CRYPTO] *Cryptography mechanisms: rules and recommendations concerning the choice and the dimensioning of the cryptographic mechanisms standard level robustness. Version 1.02, November 2004. DCSSI.*
- [ST] *TrueCrypt Security Target, Version 1.07. May 2007, SPM030-ST-1.07.*
- [ADV_FSP] *TOE functional specifications, Version 2.00 October 2007, SPM030-SP-2.00.*
- [ADV_TDS] *TOE design, Version 2.00, October 2007, SPM030-TDS-2.00.*
- [CRYPTO_SPEC] *Cryptographic specifications, Version 2.00, October 2007, SPM030_SPEC.*

付録 C： コモン・クライテリア関連の情報

C.1. 序章

本書は、ADV クラスの ADV_ARC ファミリの ADV_ARC.1 コンポーネントの要件に適合している。ここでは、TRUECRYPT ソフトウェアの「セキュリティ・アーキテクチャ」について説明する。

ADV 保証クラスは、コモン・クライテリアで定められている評価の過程において要求される、TOE の開発に関連した情報を提供する。これらの情報は、脆弱性分析やテストの実施のために使用される。

このクラスには、異なるレベルの抽象化において TSF を構造化し表現する 6 つのファミリの要件を含む。

- SFR の設計、および実装に関連の要件を含むファミリは、次の 3 つである。
 - + ADV_FSP (機能仕様)
 - + ADV_TDS (TOE 設計)
 - + ADV_IMP (実装表現)
- ドメイン分離、TSF の自己保護、セキュリティ機能のバイパス防止に関するアーキテクチャの記述に関する要件は、以下のファミリである。
 - + ADV_ARC
- セキュリティ・ポリシーモデル、および機能仕様に関する記述の要件は、以下のファミリである。
 - + ADV_SPM
- モジュールの側面、レイヤー化、及び複雑さの低減に関する TSF 内部構造の記述の要件は、以下のファミリである。
 - + ADV_INT

これらファミリのコンポーネントの目的は、以下の 2 つの特性を実証することである。

- 第一に、TOE がその仕様に基づき正確に機能していること。これに必要なファミリは、ADV_FSP、ADV_SPM、ADV_TDS、および ADV_IMP ファミリである。
- 第二に、TOE がそのセキュリティ機能をバイパスされたり、改竄されるような使用ができないこと。この特性の実証には、ADV_ARC および ADV_INT ファミリが対応する。

C.2 . ADV_ARC.1 コンポーネントの目的

このファミリの目的は、開発者にとって TSF のセキュリティ・アーキテクチャ記述を提供することである。TSF のその他の証拠資料を組み合わせこの情報を分析することによって、TSF が期待通りの特性を有していることを確認することができる。セキュリティ・アーキテクチャの記述は、TOE のセキュリティ分析は TSF の検査により達成できることを暗黙的に主張する。適切なアーキテクチャなしでは、全ての TOE の機能を検証しなければならない。

C.3. メソッドロジ

ADV_ARC では、次のような要件が必須である。

- TSF により維持されるセキュリティ・ドメインの識別とその説明
- TOE の初期化プロセスがセキュリティを保護していることの正当化
- TSF がタンパリングから自身を保護できることを保証する十分な情報が含まれていること
- SFR 実施メカニズムがバイパスされない説明

C.3.1. ドメイン分離

ドメイン分離とは、TSF は個々の信頼できないアクティブなエンティティのためのセキュリティ・ドメインを作成し、そのドメインのリソースへアクセスを許容しつつも、他のドメインに入ることができないようにドメインをお互いに分離し維持する特性である。例えば、あるオペレーティング・システム (TOE) が、信頼のおけないエンティティに関連するプロセスにドメイン (アドレス空間、プロセスごとの環境変数など) を提供することである。TOE の中には、信頼のおけないエンティティに関する全ての活動が TSF で管理されるため、そういったドメインが存在しないものもある。つまり、ドメインが存在するかどうかは、

- TOE の種別
- TOE に関連する SFR

に依存する。

C.3.2. 初期化プロセスにおけるセキュリティの維持

TOE が「動作していない」状態 (アプリケーションが開始されていない、値が初期化されていない) から「初期化」 (アプリケーションが開始されているが使用されていない、値が初期化済) の状態に移る場合には、セキュリティが維持されていなければならない。

C.3.3. 自己保護

自己保護とは、TSF に影響を及ぼす恐れがある外部エンティティによる操作から、TSF が自身を保護することができることである。こういった特性なしでは、TSF は自身のセキュリティ機能を実行することはできない。

TOE では、この保護を実現するために他の IT エンティティが提供するサービスやリソースを利用することが多い。この場合、その IT エンティティに依存し TSF が保護されるため、TSF は自身を全て保護してはいない。

C.3.4. バイパス防止

バイパス防止とは、(SFR を実現する) TSF セキュリティ機能が常に使用され、回避することができない特性である。

例えば、ファイルアクセス制御が SFR を実現する TSF の機能である場合、このアクセス制御メカニズムを経ずにファイルへアクセスすることができるインターフェースがあってはならない。

C.3.5. ADV_ARC.1 コンポーネントの要件

開発者アクションエレメント

コンポーネント CC - パート 3 (対応)	説明
ADV_ARC.1.1D	開発者は、TSFのセキュリティ特性がバイパスされないようにTOEを設計及び実装しなければならない。
ADV_ARC.1.2D	開発者は、TSFが信頼できない能動的なエンティティによって改ざんされるのを防ぐことができるようにTSFを設計及び実装しなければならない。
ADV_ARC.1.3D	開発者は、TSFのセキュリティ・アーキテクチャ記述を提供しなければならない。

内容・提示エレメント

コンポーネント CC - パート 3 (対応)	説明
ADV_ARC.1.1C	セキュリティ・アーキテクチャ記述は、TOE設計文書に記述されているSFR実施抽象概念の記述に見合った詳細レベルでなければならない。
ADV_ARC.1.2C	セキュリティ・アーキテクチャ記述は、TSFによって維持されるセキュリティ・ドメインを、SFRと一貫する形で記述しなければならない。
ADV_ARC.1.3C	セキュリティ・アーキテクチャ記述は、TSFの初期化プロセスのセキュリティがどのようにして確保されるのかを記述しなければならない。
ADV_ARC.1.4C	セキュリティ・アーキテクチャ記述は、TSFが改ざんから自分自身を保護することを実証しなければならない。
ADV_ARC.1.5C	セキュリティ・アーキテクチャ記述は、TSFがSFR実施機能性のバイパスを防ぐことを実証しなければならない。

C.3.6. CEMの基本的なタスク

CEM(評価手法)では、評価者の基本的なタスクが説明されている。これらのタスクを適切に成し遂げるには、評価者は、さまざまな評価文書の中の関連部分をベースに評価を実施しなければならない。

要求される評価文書:

- セキュリティ・ターゲット[ST]
- TOE 機能仕様[ADV_FSP]
- TOE 設計[ADV_TDS]
- セキュリティ・アーキテクチャ[ADV_ARC]
- (もし利用可能であれば) 実装表現[ADV_IMP]
- ユーザ運用マニュアル[AGD_OPE]

以下の表は、CEMのワークユニット毎に開発者から提供された対応箇所を示したものである。

コンポーネント CC - パート 3	要件 CC - パート 2	説明	開発者の対応
ADV_ARC.1.1C	セキュリティ・アーキテクチャ記述は、TOE 設計文書に記述されている SFR 実施抽象概念の記述に見合った詳細レベルでなければならない。		
	ADV_ARC.1-1	評価者は、証拠で提供されている情報が、機能仕様と TOE 設計文書に含まれている SFR 実施抽象概念の記述に見合った詳細レベルで提示されていることを決定するために、セキュリティ・アーキテクチャ記述を検査しなければならない。	ADV_TDS では、TOE がサブシステム、およびモジュールとして記述されている。本書の (TOE の) 説明は、ADV_TDS に記載されている説明と同レベルである。
ADV_ARC.1.2C	セキュリティ・アーキテクチャ記述は、TSF によって維持されるセキュリティ・ドメインを、SFR と一貫する形で記述しなければならない。		
	ADV_ARC.1-2	評価者は、TSF によって維持されるセキュリティ・ドメインをセキュリティ・アーキテクチャ記述が記述していることを決定するために、その記述を検査しなければならない。	ドメイン分離の考え方は、13 ページの第 2 章「TSF が維持するセキュリティ・ドメイン」で説明されている。
ADV_ARC.1.3C	セキュリティ・アーキテクチャ記述は、TSF の初期化プロセスのセキュリティがどのようにして確保されるのかを記述しなければならない。		
	ADV_ARC.1-3	評価者は、初期化プロセスのセキュリティが保持されていることを決定するために、セキュリティ・アーキテクチャ記述を検査しなければならない。	TSF の初期化におけるセキュリティ維持の考え方は、14 ページの第 3 章「TOE の初期化におけるセキュリティ保護」で説明されている。
ADV_ARC.1.4C	セキュリティ・アーキテクチャ記述は、TSF が改ざんから自分自身を保護することを実証しなければならない。		
	ADV_ARC.1-4	評価者は、セキュリティ・アーキテクチャ記述が、信頼できない能動的なエンティティによる改ざんから TSF が自分自身を保護できるといふ決定を支持するのに十分な情報を含んでいることを決定するために、その記述を検査しなければならない。	TSF の自己保護は、21 ページの第 4 章「」で説明されている。
ADV_ARC.1.5C	セキュリティ・アーキテクチャ記述は、TSF が SFR 実施機能性のバイパスを防ぐことを実証しなければならない。		
	ADV_ARC.1-5	評価者は、SFR 実施メカニズムをバイパスできないようにするしくみを適切に説明する分析をセキュリティ・アーキテクチャ記述が提示していることを決定するために、その記述を検査しなければならない。	TSF がセキュリティ・メカニズムのバイパス防止している根拠は、23 ページの第 5 章「SFR 実施メカニズムのバイパス防止」で説明されている。