TOSHIBA

東芝ソリューション暗号ライブラリ
セキュリティポリシ
Ver.1.0.1

2007 年 03 月 28 日

東芝ソリューション株式会社
IT 技術研究所

更新履歴

Version	日付	変更内容
0.9.0	2006/09/29	新規作成
0.9.1	2007/01/29	試験機関の所見に基づく修正
1.0.0	2007/02/19	試験機関の所見に基づく再修正
1.0.1	2007/03/28	認証機関の所見に基づく再修正

目　次

1. はじめに

1.1. 目的

本文書は、東芝ソリューション暗号ライブラリに関するセキュリティポリシである。本文書は電子政府推奨暗号リスト等に示された暗号アルゴリズムの実装に関する試験等を行う暗号モジュール試験及び認証制度（以下「JCMVP」という）で要求される文書の１つとして提供される。

1.2. 本文書の位置づけ

東芝ソリューション暗号ライブラリのパッケージは以下のもので構成されている。

(1) セキュリティポリシ
以下のものを含む。
- 有限状態モデル
- ブロック図

(2) 利用ガイダンス
以下のものを含む。
- クリプトオフィサガイダンス
- ユーザガイダンス

(3) API 仕様書
(2)の利用ガイダンスに付属する文書である。

(4) 暗号ライブラリ本体（DLL）
暗号モジュール本体であり、(1)(2)(3)はこれに関して記述している。

これら一覧の中で、本文書は(1)に該当するものである。

2. 暗号モジュールの仕様

2.1. 暗号モジュール概要

東芝ソリューション暗号ライブラリは、Microsoft Windows OS 上で動作するダイナミックリンクライブラリ（DLL）である。
以下「暗号モジュール」もしくは「暗号ライブラリ」という場合は、この東芝ソリューション暗号ライブラリを指すものとする。
この暗号モジュールは、FIPS140-2 の言葉でいえば、マルチチップスタンドアロン型暗号モジュールである。
本暗号モジュールは、JCMVP におけるセキュリティレベル１の試験対象としている。
また本暗号モジュールのバージョンは 1.0.0.0 である。

2.2. 暗号境界

本暗号モジュールの物理的な境界は、暗号モジュールを動作させるコンピュータ全体の境界である。
本暗号モジュールの論理的な境界は、暗号ライブラリの機能全体の境界である。

2.3. ハードウェアプラットフォーム

本暗号モジュールは表 2-1に示すハードウェアプラットフォームで動作する。

表 2-1 ハードウェア環境

CPU	メモリ	HDD
Pentium 4 800MHz 以上	512MB 以上	1GB 以上

2.4. ブロック図

本暗号モジュールのブロック図を図 2-1に示す。また、この中に暗号境界と入出力ポート（I/O ポート）を示している。

入出力ポートは、

- 入力物理ポート：キーボードポート、マウスポート、ネットワークポート
- 出力物理ポート：モニタポート、ネットワークポート

である。

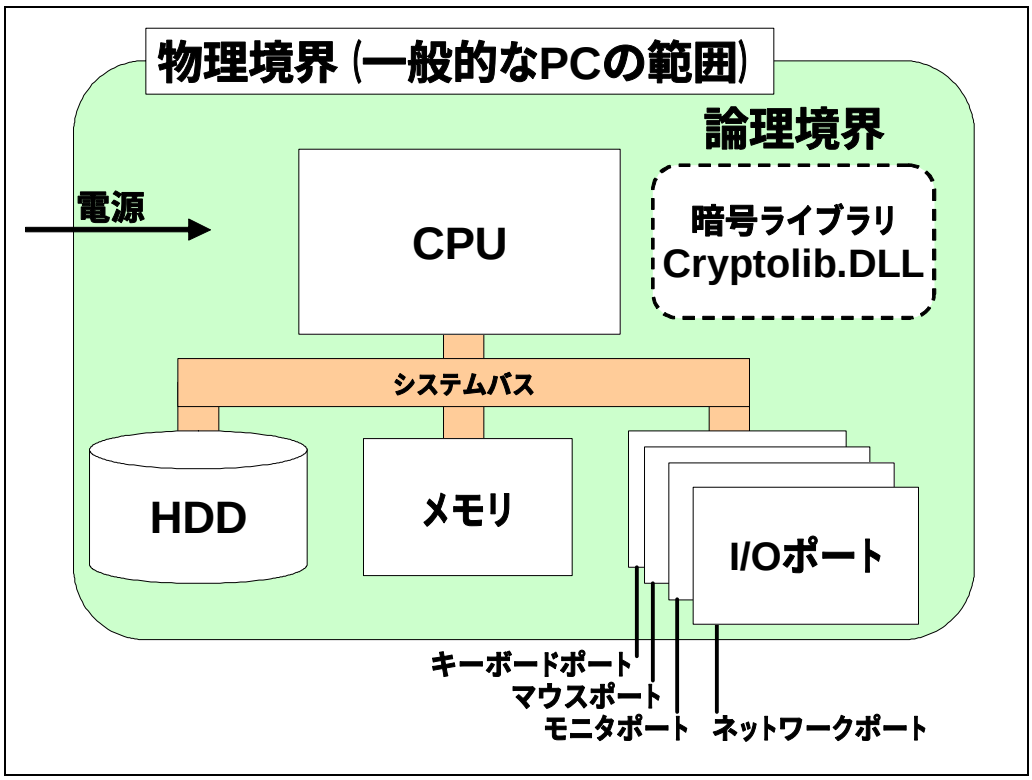

図 2-1 暗号ライブラリのブロック図

2.5. ソフトウェア環境

本暗号モジュールは次のソフトウェア環境で動作する。

OS：Microsoft Windows XP SP2 Professional Edition

また、本暗号モジュールは次のソフトウェア環境でビルドし、ダイアミックリンクライブラリとして生成されている。

OS：Microsoft Windows XP SP2 Professional Edition
開発ツール：Microsoft Visual C++ 2005 Express Edition

2.6. 承認された動作モード

本暗号モジュールは、承認されたセキュリティ機能８個を実装している。そのリストを表 2-2 に示す。
これらのセキュリティ機能は通常の関数と同じように、他のモジュールから API を指定することによって呼び出して、使用することが可能である。

表 2-2 承認されたセキュリティ機能

カテゴリ	アルゴリズム名
128 ビットブロック暗号	AES
64 ビットブロック暗号	3-key Triple DES
ハッシュ	SHA-1
	SHA-256
署名	RSASSA-PSS
メッセージ認証	HMAC-SHA-1
	HMAC-SHA-256
乱数生成	PRNG based on SHA-1 for general purpose in FIPS 186-2 (+ change notice 1) revised Appendix 3.1

本暗号モジュールに実装されたセキュリティ機能は、全て承認されたセキュリティ機能であるため、常に承認された動作モードである。承認されたセキュリティ機能は、4.2節に示している各サービスとして実装されている。ユーザは状態遷移図での動作可能状態に移行することで、4.2節に示している各サービスを利用することが可能になる。
状態遷移図での動作可能状態に移行していない状態では、4.2節で示しているどのサービスも利用することはできない。従って、状態遷移図での動作可能状態に移行していない状態で、本モジュールで提供するサービスを利用しても出力データを正しく得ることはできない。

3. ポートとインターフェース

本モジュールは、API を通じて論理的なインターフェースを提供する。この論理的なインターフェースは本モジュールを用いるアプリケーション等が利用する。
本モジュールで提供される API インターフェースは、以下の論理的インターフェースに分類される。

表 3-1　インターフェース

インターフェース	論理的インターフェース	物理的ポート
データ入力インターフェース	本モジュールの関数で処理するためのデータを入力するデータパス	標準的な入力ポート（例：キーボード）
データ出力インターフェース	本モジュールの関数で処理したデータを出力するデータパス	標準的な出力ポート（例：モニター）
制御入力インターフェース	本モジュールにサービスを提供する全ての関数	N/A
状態出力インターフェース	本モジュールの状態情報を出力するデータパス	標準的な出力ポート（例モニター）

本モジュールにおけるデータ入力インターフェースとデータ出力インターフェースのデータの流れは API で論理的に分離されている。
メッセージダイジェストを作成する SHA（SHA-1,SHA-256）の Update 関数では、データ入力インターフェースとデータ出力インターフェースは同じであるが、関数内のメッセージダイジェスト更新処理で入力データを別の変数に格納して更新処理を行っているので、論理的には分離されており、入力インターフェースと出力インターフェースとの間でデータの混合は起こらない。HMAC（HMAC/ SHA-1,SHA-256）の Update 関数についても、関数内で SHA の Update 関数を呼んでいるので、同様に入力インターフェースと出力インターフェースの間でデータの混合は起こらない。

4. 役割、サービス、及び認証

4.1. 役割

本モジュールでは、ユーザ役割とクリプトオフィサ役割がサポートされる。各役割の内容は以下の通り。

ユーザ役割	ユーザ役割は、本モジュールで提供される API を用いた暗号動作を行う場合に暗黙的に割り当てられる役割であり、表 4-1のユーザ役割の各 API にアクセス可能である。
クリプトオフィサ役割	クリプトオフィサ役割は、本モジュールのコンピュータへの導入作業、オペレーティングシステムの設定変更、本モジュールで提供されるあらゆる API にアクセス可能である。

なお、本モジュールでは、メンテナンス役割をはじめとする上記 2 役割以外の役割はサポートされない。

4.2. サービス

本モジュールで提供されるサービスを以下の表にまとめる。

表 4-1　提供されるサービスと役割

サービス	アルゴリズム名	規格	API	役割
共通鍵暗号	AES	FIPS 197	aes_setkey aes_ecb_encrypt aes_ecb_decrypt aes_cbc_encrypt aes_cbc_decrypt	**ユーザ役割** **クリプトオフィサ役割**
	3-key Triple DES	FIPS 46-3	tdes_setkey tdes_ecb_encrypt tdes_ecb_decrypt tdes_cbc_encrypt tdes_cbc_decrypt	**ユーザ役割** **クリプトオフィサ役割**
デジタル署名	RSASSA-PSS	PKCS#1 v2.1	rsassa_pss_sign_sha1 rsassa_pss_verify_sha1 rsassa_pss_sign_sha256 rsassa_pss_verify_sha256	**ユーザ役割** **クリプトオフィサ役割**
メッセージダイジェスト	SHA-1	FIPS 180-2	sha1Init sha1Update sha1Final sha1All	**ユーザ役割** **クリプトオフィサ役割**
	SHA-256	FIPS 180-2	sha256Init sha256Update sha256Final sha256All	**ユーザ役割** **クリプトオフィサ役割**
	HMAC/SHA-1,SHA-256	FIPS 198	HMAC_sha1Init HMAC_sha256Init HMAC_Update HMAC_Final HMAC_sha1 HMAC_sha256	**ユーザ役割** **クリプトオフィサ役割**
乱数生成	PRNG based on SHA-1 for general purpose in FIPS 186-2 (+ change notice 1) revised Appendix 3.1	FIPS 186-2	random_generate	**ユーザ役割** **クリプトオフィサ役割**
			pseudo_random_generate set_seed set_xkey	**クリプトオフィサ役割**
その他	**自己テスト**	N/A	PowerUpSelfTest	**ユーザ役割** **クリプトオフィサ役割**
	状態の表示	N/A	get_last_error_code get_last_error_string	**ユーザ役割** **クリプトオフィサ役割**
	RSA **鍵初期化/解放**	N/A	rsa_privatekey_init rsa_privatekey_clear rsa_publickey_ init rsa_publickey_clear	**ユーザ役割** **クリプトオフィサ役割**

4.3. 認証

本モジュールは、役割を区別する認証手段をもたない。役割は利用するサービスにより暗黙的に区別される。

4.4. 乱数生成について

乱数生成として提供される random_generate **と** pseudo_random_generate **のサービスの違いは次の通りである。**

pseudo_random_generate **は、**FIPS 186-2 **に従うアルゴリズムを忠実に実装したもので、これを用いて乱数を生成するためには、これに先立ってシード、シード鍵をそれぞれ** set_seed, set_xkey **を用いて設定することが必要である。**pseudo_random_generate, set_seed, set_xkey **はクリプトオフィサ向けサービスとして提供される。**

一方、random_generate **は、**set_seed, set_xkey, pseudo_random_generate **を順に呼び出して、生成された乱数に対して連続乱数生成器テストを行い、合格した乱数を出力する。**

random_generate **は、一般のユーザ向けサービスとして提供される。**

5. 有限状態モデル

5.1. 状態遷移図

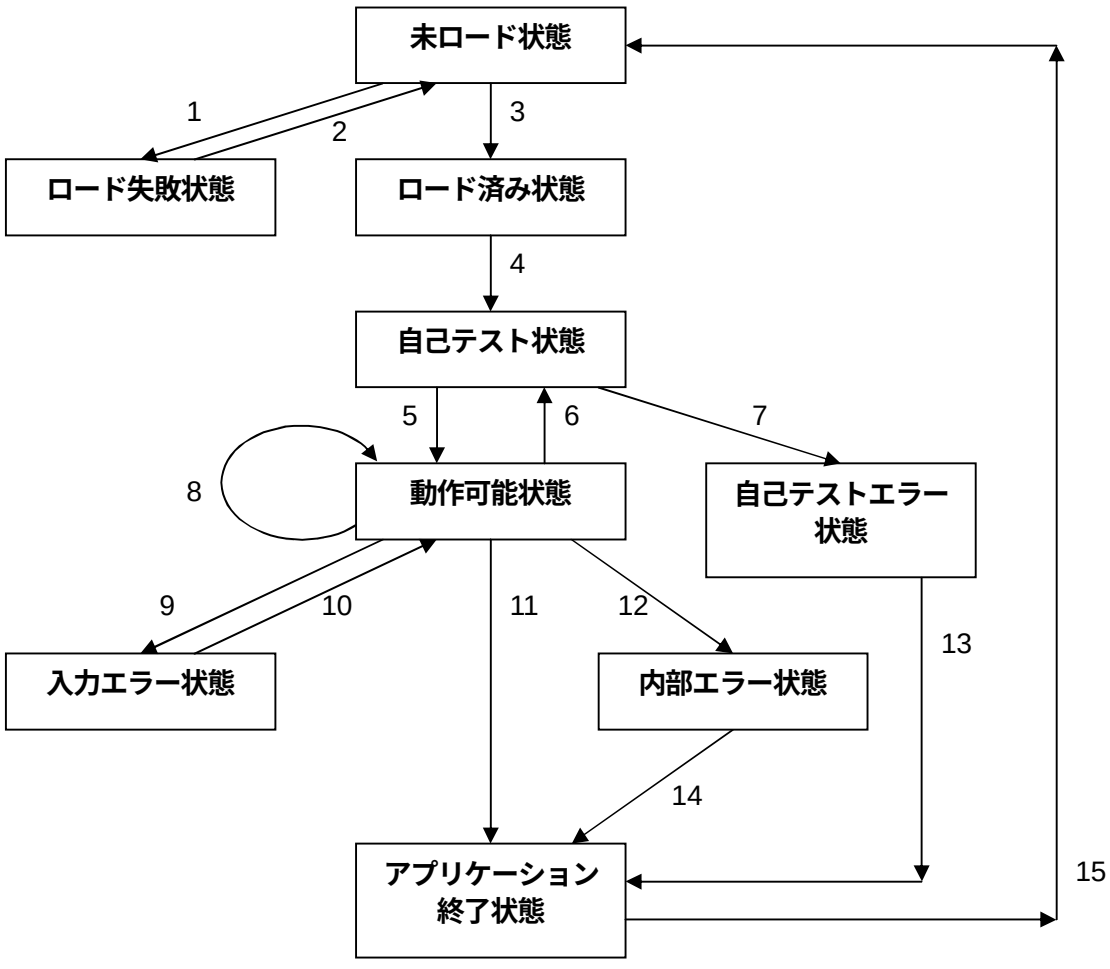

図 5-1 有限状態モデル図

5.2. 状態の説明

5.2.1. 未ロード状態

モジュールがホストとなるオペレーティングシステム上のメモリ上にロードされていない状態。

5.2.2. ロード済み状態

モジュールがホストとなるオペレーティングシステム上のメモリ上にロードされている状態。未ロード状態からユーザのロード操作が成功すると遷移する。この状態ではモジュールがオペレーティングシステム上で動作しているが、まだパワーアップ自己テストを行っていない状態。電源 ON 状態に相当する。

5.2.3. ロード失敗状態

モジュールをホストとなるオペレーティングシステム上のメモリ上にロードする際にエラーが生じた状態。未ロード状態からユーザのロード操作が失敗すると遷移する。

5.2.4. 自己テスト状態

自己テストを行っている状態。ロード済み状態後のパワーアップ自己テスト、その後の動作可能状態からオペレータからのオンデマンドで行うパワーアップテスト、条件自己テストのいずれかを行っている状態。

5.2.5. 自己テストエラー状態

自己テスト状態で、1 つでも自己テストがエラーとなった場合に遷移する状態。この状態から遷移できる状態はアプリケーション終了状態のみである。

5.2.6. 動作可能状態

モジュールがすべての自己テストで合格となり、アプリケーションからの呼び出しを待っている状態。ユーザ状態に相当する。

5.2.7. 入力エラー状態

アプリケーションから入力されたデータが起因となるエラー（データフォーマット異常など）が発生した状態。自動的に動作可能状態に遷移する。

5.2.8. 内部エラー状態

暗号ライブラリ内の不具合によるエラーが発生した状態。この状態から遷移できる状態はアプリケーション終了状態のみである。

5.2.9. アプリケーション終了状態

ユーザがアプリケーションを終了した状態。電源 OFF 状態に相当する。

5.3. 状態遷移表

表 5-1 状態遷移のトリガとなる入力と遷移後の出力

	現在の状態	入力	出力	次の状態
1	未ロード	ロード失敗	（ロード失敗メッセージ：アプリケーションからの出力）	ロード失敗
2	ロード失敗	自動遷移	なし	未ロード
3	未ロード	ロード成功	（ロード成功メッセージ：アプリケーションからの出力）	ロード済み
4	ロード済み	自動遷移	自己テスト開始	自己テスト
5	自己テスト	自己テスト終了	自己テスト成功	動作可能
6	動作可能	自己テストオンデマンド実行、および条件自己テスト	自己テスト開始	自己テスト
7	自己テスト	自己テスト終了	自己テスト失敗	自己テストエラー
8	動作可能	モジュール API 呼び出し	正常終了	動作可能
9	動作可能	入力値異常でのモジュール API 呼び出し	入力値エラーメッセージ	入力エラー
10	入力エラー	自動遷移	なし	動作可能
11	動作可能	アプリケーション終了操作	モジュールのアンロード	アプリケーション終了
12	動作可能	内部関数がエラーを出力	内部関数エラーメッセージ	内部エラー
13	自己テストエラー	アプリケーション終了操作	モジュールのアンロード	アプリケーション終了
14	内部エラー	アプリケーション終了操作	モジュールのアンロード	アプリケーション終了
15	アプリケーション終了	自動遷移	なし	未ロード

6. 物理的セキュリティ

2で述べたように本暗号モジュールは、ＰＣ上で動作するソフトウェアである。これは FIPS140-2 の言葉で言うと、マルチチップスタンドアロン型暗号モジュールであって、FIPS140-2 で要求されるレベル１の物理的セキュリティの要求は満たしている。

7. 動作環境

7.1. オペレーティングシステム

レベル１の暗号モジュールにおいては、オペレーティングシステムに対して単一オペレータ動作モードが要求される。対象とするオペレーティングシステムにおいて、本モジュールを利用するユーザのアプリケーションはオペレーティングシステムの制御下でプロセスに割り付けられ動作する。各プロセスは物理的なメモリ空間とは論理的に分離された仮想アドレス空間のメモリを参照した実行スレッドにより動作する。アプリケーションはプロセス単位

で管理され、本モジュールはプロセス間通信を使用しないため、個々のプロセスごとに独立して動作しており、他のプロセスからモジュール内の暗号鍵および CSP にアクセスできない。また、他のプロセスによる割り込みもできない。したがって、JCMVP における単一オペレータ動作モードの制限下で動作する。

7.2. 完全性

DLL モジュールのロード時のパワーアップ自己テストにより、モジュールの完全性が検査される。
完全性検査の手段には、モジュールに内蔵されたモジュール完全性検証専用の公開鍵を参照して RSASSA-PSS によるデジタル署名の検証を用いる。

7.3. その他

本モジュールをマルチスレッドアプリケーションでの利用を制限しない。ただし、利用者は以下の点に留意すること。

- モジュールはスレッド同期の手段を提供しない。
- Win32 DisableThreadLibraryCalls API を呼び出してはならない。スレッド通知メッセージを無効にして生成したスレッドはメモリ確保失敗によるエラー状態を返す。

また、安全性を考慮し、RSA を使用する場合は鍵長 1024 ビット以上の鍵を用いることを推奨する。但し、本ライブラリで RSA の鍵を扱う場合、鍵長は 4096 ビット以下でないと動作を保証することはできないので注意が必要である。

8. CSP 管理

8.1. CSP の種類

本暗号モジュールで管理している CSP はモジュール内部で生成した乱数生成器のシード鍵のみである。
暗号モジュールに用いられる全ての暗号鍵、暗号鍵コンポーネントは暗号モジュールの論理境界の外で管理されている。

8.2. CSP の管理

シード鍵は、本モジュールでは CSP として次のように管理している。本モジュール内の乱数生成の関数を呼び出すと、乱数生成を行った後にシード鍵をゼロ化している。また、乱数生成の関数実行中においてシード鍵はオペレーティングシステムのメモリ上に平文で格納され、そのメモリはオペレーティングシステムの保護機能によって不正なアクセスから保護される。
乱数は、RSA-PSS の署名生成時における SALT としても利用している。この SALT も使用後にはゼロ化される。

承認されたモードであってもなくても、ユーザが4.2節のクリプトオフィサ役割でしか利用できないサービスを利用してはならない。もし利用した場合の本モジュールの動作および本モジュールを用いて構成されるアプリケーションの動作については保証できない。

9. 自己テスト

本暗号ライブラリは、完全性と承認されたセキュリティ機能の正常な動作を確認するため、以下に述べるパワーアップ自己テスト及び条件自己テストの機能を備えている。

なお、暗号ライブラリが持つセキュリティメカニズムはこれら自己テストだけであり、暗号ライブラリを改竄する攻撃に対処することが可能である。

9.1. パワーアップ自己テスト

パワーアップ自己テストはアプリケーションのプロセスに DLL モジュールが読み込まれる実行スレッドが開始される際に自動的に行われる。オペレータによるいかなる入力も必要としない。パワーアップ自己テストでは、後述のソフトウェア完全性テストと暗号アルゴリズムテストが実行される。実行結果に異常が検出されたならば、対応するエラーコードの TSCL_INTEGRITY_TEST_FAILED または TSCL_KAT_FAILED を返し、そのプロセスの実行スレッドはエラー状態となりいかなる暗号動作も実行できない。

9.1.1. 暗号アルゴリズムテスト

暗号アルゴリズムテストは既知解テスト（Known Answer Tests、KAT と略す）により実行される。既知解テストを行う暗号アルゴリズムを表 9-1に示す。既知解テストでは、各暗号アルゴリズムから出力される値が、事前にモジュール内部に格納されている値と比較される。その比較結果が異なる場合は、既知解テスト失敗としてモジュールはエラーコード TSCL_KAT_FAILED を返し、自己テストエラー状態に遷移する。

表 9-1 既知解テストを実行するアルゴリズムと方法

アルゴリズム	既知解テストの方法
AES	128 ビット鍵による ECB モード暗号化
	128 ビット鍵による ECB モード復号
	128 ビット鍵による CBC モード暗号化
	128 ビット鍵による CBC モード復号
3-key Triple DES	192 ビット鍵による ECB モード暗号化
	192 ビット鍵による ECB モード復号
	192 ビット鍵による CBC モード暗号化
	192 ビット鍵による CBC モード復号
SHA-1	SHA-1 メッセージダイジェストの生成
SHA-256	SHA-256 メッセージダイジェストの生成
RNG	既知シードによる擬似乱数生成
HMAC	160 ビット鍵による HMAC-SHA-1 メッセージダイジェストの生成
	160 ビット鍵による HMAC-SHA-256 メッセージダイジェストの生成
RSASSA-PSS	ハッシュ関数=SHA-1、1024 ビット鍵による署名生成
	ハッシュ関数=SHA-1、1024 ビット鍵による署名検証
	ハッシュ関数=SHA-256、1024 ビット鍵による署名生成
	ハッシュ関数=SHA-256、1024 ビット鍵による署名検証

9.1.2. ソフトウェア完全性テスト

ソフトウェア完全性テストでは、本モジュールのデジタル署名データを除く読み込み専用領域のロードイメージダイジェストを用いて DLL 内部のデジタル署名の検証を行い、完全性を保証する。デジタル署名のアルゴリズムは SHA-256 を用いた RSASSA-PSS である。デジタル署名の検証結果が異常な場合は、ソフトウェア完全テスト失敗としてモジュールはエラーコード TSCL_INTEGRITY_TEST_FAILED を返し、自己テストエラー状態に遷移

する。

9.1.3. 自己テストのオンデマンド実行

PowerUpSelfTest の呼び出しにより、自己テストのオンデマンド実行が可能である。自己テストはソフトウェア完全性テストと暗号アルゴリズムテストを実行し、これらテストがエラーを返したならば、モジュールは対応するエラーコードの TSCL_INTEGRITY_TEST_FAILED または TSCL_KAT_FAILED を返し、呼び出したプロセスの実行スレッドは自己テストエラー状態に遷移する。復帰するための唯一の手段は、ライブラリを明示的にアンロードあるいはプロセスの実行スレッドを破棄し暗黙的にアンロードし、再ロードすることである。

9.2. 条件自己テスト

9.2.1. 連続乱数生成器テスト

連続乱数生成器テストは、生成した乱数の上位 20 バイトを、前回生成された乱数の上位 20 バイトと比較することで実行される。電源投入後等、前回生成された乱数が存在しない場合は、まず乱数生成を 1 回実行してその結果の上位 20 バイトを前回生成された乱数として保存した後、続けて乱数生成をもう 1 回実行し、そこで生成された乱数の上位 20 バイトを 1 回目に生成された乱数の上位 20 バイトと比較することで実行される。その結果、前回生成された乱数 20 バイトと異なる場合は、その結果を乱数として出力する。前回生成された乱数 20 バイトと一致した場合は、連続乱数生成器テスト失敗としてモジュールはエラーコード TSCL_CONTINUOUS_RNG_TEST_FAILED を返し、自己テストエラー状態に遷移する。

10. 設計保証

10.1. 構成管理

暗号境界内の暗号モジュールに関係する構成要素は

- ソースコード
- モジュール（DLL 形式）
- セキュリティポリシ
- 利用ガイダンス

である。このうちユーザおよびクリプトオフィサに提供されるものは、モジュール、セキュリティポリシ、利用ガイダンスである。

ソースコードはバージョン管理ソフト Subversion（http://subversion.tigris.org/）により管理されている。Subversion に格納された各ファイルは、バージョンごとに一意的に識別できる情報がつけられて管理される。したがって、各ファイルのバージョンは Subversion のクライアントソフトウェアを通すことで、一意的に識別可能となっている。
Subversion に格納された各ファイルは、限定された開発者のみに修正を許可するようなアクセス制御を行っている。
Subversion のリポジトリは、モジュールのバージョンごとにブランチが切られる構成となっている。逆に言えばモジュールのバージョンは Subversion のリポジトリのブランチで管理されている。モジュール自体にもバージョン情報は記録され、Windows 上のファイルとしてのプロパティを見ることで、バージョンを確認できる。
セキュリティポリシ、利用ガイダンスも該当ブランチに格納することで、バージョン管理を行っている。

10.2. 配布及び運用

暗号モジュール、利用ガイダンスは CD-ROM による配布のみとしている。セキュリティポリシについては、CD-ROM による配布の他、JCMVP の認証を取得した暗号モジュールのセキュリティポリシとして、Web でも公開される予定である。
なお、暗号モジュールの SHA-256 によるダイジェストは表 10-1の通りである。

表 10-1　配布ファイルのダイジェスト

対象	ダイジェスト（１６進表示）
暗号モジュール	1E:86:05:09:BF:A9:36:E9:4D:1C:A8:B4:24:27:FB:D0:F3:CE:FF:96:22:99:9F:C5:74:A7:A5:AC:89:31:D7:36

CD-ROM 内のファイルに対する SHA-256 のダイジェストが、上記の値に一致するか検証することにより CD-ROM 内のファイルが変化していないことを確認できる。
また CD-ROM 自体には発行番号を印字し、配布先と発行番号のペアで管理している。発行番号は、配布元のみが知っているアップデート関数を利用して順次発行することで、第三者に予測できないようにしている。
モジュールの初期設定（インストール）に関しては、利用ガイダンスの 2 章を参照のこと。

10.3. ガイダンス文書

1.2で述べた利用ガイダンスの中の「クリプトオフィサガイダンス」にて本暗号モジュールの入手、完全性確認方法、インストール方法について説明し、利用ガイダンスの中の「ユーザガイダンス」と「ＡＰＩ仕様」にて本暗号モジュールの使用方法を説明している。

11. 参考文献

[1] FIPS 197 Advanced Encription Standard (AES)
[2] FIPS 46-3 Data Encryption Standard (DES)
[3] National Institute of Standards and Technology, Security Requirements for Cryptographic Modules. FIPS 140-2, 25 May, 2001.

[4] National Institute of Standards and Technology, Digital Signature Standard (DSS), FIPS 186-2, October 5, 2001.

[5] National Institute of Standards and Technology, Secure Hash Standard, FIPS 180-2, August 1, 2002.
[6] National Institute of Standards and Technology, Keyed-Hash Message Authentication Code (HMAC) FIPS 198, issued March 6, 2002.
[7] RSA Laboratories. PKCS #1: RSA Encryption Standard. Version 2.1, June 14, 2002.
[8] National Institute of Standards and Technology, Recommendation for Key Management – Part 1: General, NIST Special Publication 800-57 (Draft), April, 2005.