Keymate/Crypto JCMVP ライブラリ (Solaris 版及び Windows 版) セキュリティポリシ

バージョン：05

2008 年 11 月 25 日

(株)日立製作所ソフトウェア事業部

変更履歴

バージョン	日付	更新内容
00	2008/9/3	新規作成
01	2008/9/19	試験機関の所見に基づく改訂
02	2008/10/10	試験機関の所見に基づく改訂
03	2008/10/23	試験機関の所見に基づく改訂
04	2008/10/28	試験機関の所見に基づく改訂
05	2008/11/25	試験機関の所見に基づく改訂

目次

１はじめに

1.1 目的

本文書は，日立製作所Keymate/Crypto JCMVPライブラリ(Solaris[1]版及びWindows®[2]版)に関するセキュリティポリシである。JIS X 19790「暗号モジュールのセキュリティ要求事項」におけるセキュリティレベル１ の要求を満たすことを示す文書の１つである。以下「本暗号モジュール」は，このKeymate/Crypto JCMVPライブラリ(Solaris版及びWindows版)を指すものとする。

1.2 識別情報

名称：Keymate/Crypto JCMVP ライブラリ(Solaris 版及び Windows 版) セキュリティポリシ

識別名：P-9D44-1541 04-00 SP

作成者：(株) 日立製作所

２暗号モジュールの仕様

2.1 暗号モジュール概要

本暗号モジュールは，共有ライブラリまたは動的リンクライブラリである。本暗号モジュールは，JIS X 19790 におけるセキュリティレベル１の試験対象である。本暗号モジュールは，Keymate/Crypto製品に同梱する[3]。本暗号モジュールのバージョンは，04-00 である。

2.2 暗号境界

本暗号モジュールの暗号境界として，その物理的境界と論理的境界を述べる。本暗号モジュールの物理的境界は,本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境界は，本暗号モジュール機能全体の境界である。

2.3 プラットフォーム

(1) Solaris 版動作環境

本暗号モジュールは以下に示すプラットフォームで動作する。

CPU：SPARC プロセッサ

動作 OS：Solaris10 5/08(SPARC)

---

[1] Solaris は，米国 Sun Microsystems, Inc. の米国及びその他の国における商標または登録商標です。

[2] Windows は，米国及びその他の国における米国 Microsoft Corp.の登録商標です。

[3] Keymate/Crypto 製品には，Keymate/Crypto Development Kit Version 4 と Keymate/Crypto Run Time Version 4 がある。前者は，アプリケーションプログラムをコンパイルし実行するための製品であり，後者は前者のアプリケーションを動作させるための実行環境です。

提供形態：共有ライブラリ

(2) Windows®版動作環境

CPU：AMD[4]/Intel[5] 64 プロセッサ

動作OS:Windows Server[6]® 2008 Standard (x64), Windows Server® 2008 Enterprise (x64) それぞれ日本語版を対象とする。

提供形態：動的リンクライブラリ

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。この中に暗号境界(物理的境界と論理的境界)と入出力ポートを示している。

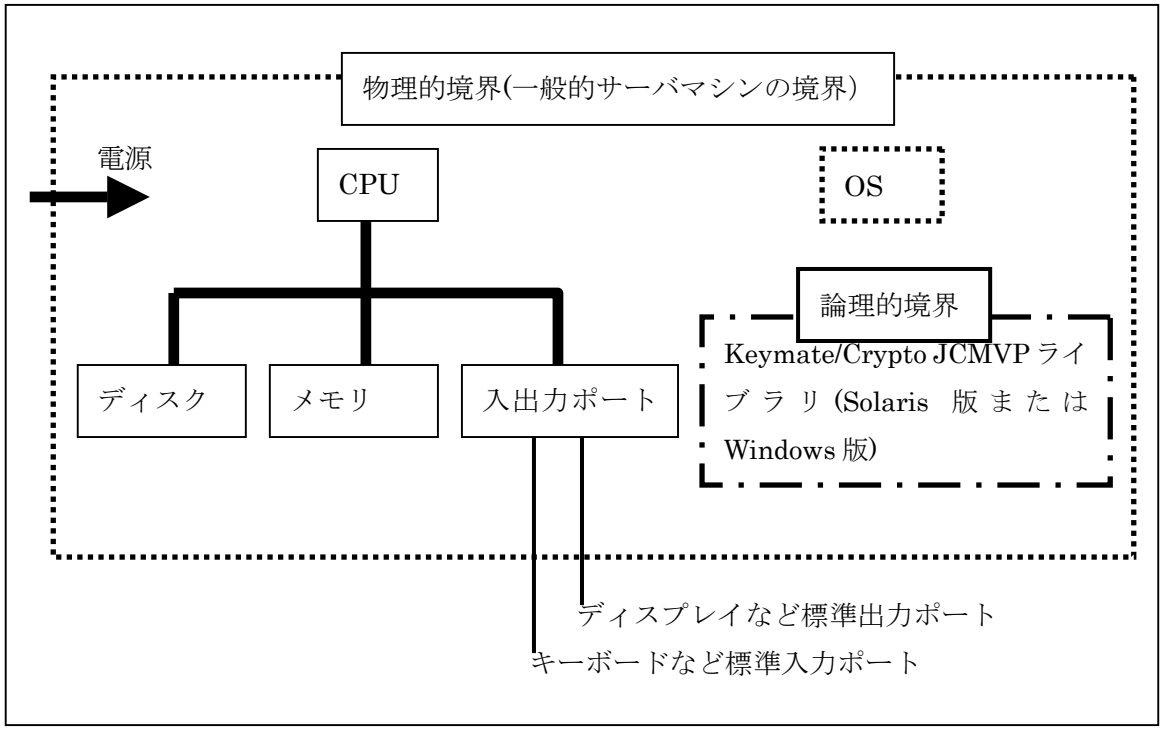

図 2-1 本暗号モジュールのブロック図

[4] AMD は，Advanced Micro Devices, Inc.の商標です。
[5] Intel は，Intel Corporation の会社名です。
[6] Windows Server は，米国 Microsoft Corporation の米国及びその他の国における登録商標です。

３ポートとインタフェース

本暗号モジュールは，API を通じて論理的なインタフェースを提供する。この論理的なインタフェースは本暗号モジュールを用いるアプリケーション等が利用する。

本暗号モジュールで提供される API インタフェースは，以下４つの論理的インタフェースを持つ。

（１） データ入力

（２） データ出力

（３） 制御入力

（４） 状態出力

インタフェースの種別,物理的ポート及び論理的インタフェースの対応を表 3-1 に示す。

表 3-1 インタフェースの種別，物理的ポート及び論理的インタフェースの対応

インタフェースの種別	物理的ポート	論理的インタフェース
データ入力	キーボードなど標準入力ポート	API 関数の入力パラメタ
データ出力	ディスプレイなど標準出力ポート	状態取得関数と自己テスト関数を除く API 関数の出力パラメタ
制御入力	キーボードなど標準入力ポート	API 関数
状態出力	ディスプレイなど標準出力ポート	API 関数の戻り値，状態取得関数の出力パラメタ，及び自己テスト関数の出力パラメタ

４役割，サービス，及び認証

4.1 役割

本暗号モジュールでは，ユーザ役割とクリプトオフィサ役割をサポートする。役割は暗黙的に割り当てる。

ユーザ役割：本暗号モジュールで提供する API を用いた暗号動作を行う役割であり，本暗号モジュールで提供するすべてのサービスを API を通して利用可能である。

クリプトオフィサ役割：コンピュータに本暗号モジュールの組込み及び削除作業を行う役割であり，本暗号モジュールで提供するすべてのサービスを API を通して利用可能である。

なお、本暗号モジュールは，メンテナンス役割はサポートしない。

4.2 サービス

本暗号モジュールで提供するサービスを示す。本暗号モジュールは JCMVP で承認されたセキュリティ機能 [1]のみを提供する。このため，本暗号モジュールは常に承認された動作モードで動作する。提供するサービスを表 4-1 に示す。

表 4-1 提供するセキュリティ機能

サービスのタイプ	アルゴリズム	仕様	備考
公開鍵署名	ECDSA	IEEE P1363 [2], SEC1 [3]	楕円曲線の定義体及びベースポイントの位数は 160 ビット以上。
	RSASSA-PKCS1-v1_5	PKCS#1v2.1 [4] (モジュラスは2つの素数の積)	モジュラスとなる合成数は 1024 ビット以上。
	RSASSA-PSS	PKCS#1v2.1 (モジュラスは2つの素数の積)	モジュラスとなる合成数は 1024 ビット以上。
公開鍵守秘	RSA-OAEP	PKCS#1v2.1 (モジュラスは2つの素数の積)	モジュラスとなる合成数は 1024 ビット以上。
共通鍵 128 ビットブロック暗号	AES	FIPS197 [5]	ECB, CBC [9]
ハッシュ関数	SHA-256	FIPS180-2 [6]	
	SHA-384	FIPS180-2	
	SHA-512	FIPS180-2	
メッセージ認証	HMAC	FIPS198a [7]	ハッシュ関数は，SHA-256, SHA-384, SHA-512
乱数ビット列生成器	Hash_DRBG (ベンダ自己確認)	SP800-90 [8]	ハッシュ関数は SHA-512
自己テスト			
状態の取得			
動作モードの取得			

4.3 認証

本暗号モジュールは，役割を識別する認証メカニズムをもたない。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 CSP及びPSP

本暗号モジュールで提供するサービスにおいて，CSP 及び PSP，これらへアクセスするタイプを表 4-2 に示す。

表 4-2 CSP 及び PSP，これらへアクセスするタイプ

サービスのタイプ	アルゴリズム	CSP	PSP	アクセスタイプ
公開鍵署名	ECDSA	プライベート鍵	公開鍵，システムパラメタ(システム鍵)	read/write
	RSASSA-PKCS1-v1_5	プライベート鍵	公開鍵	read/write
	RSASSA-PSS	プライベート鍵	公開鍵	read/write
公開鍵守秘	RSA-OAEP	プライベート鍵	公開鍵	read/write
共通鍵128 ビットブロック暗号	AES	秘密鍵(共通鍵)	なし	read/write
ハッシュ関数	SHA-256	なし	なし	-
	SHA-384	なし	なし	-
	SHA-512	なし	なし	-
メッセージ認証	HMAC	秘密鍵(共通鍵)	なし	read/write
乱数ビット列生成器	Hash_DRBG	内部状態(シード)	なし	read/write
自己テスト		なし	なし	-
状態の取得		なし	なし	-
動作モードの取得		なし	なし	-

５物理的セキュリティ

本暗号モジュールは，SPARC プロセッサまたは AMD/Intel 64 プロセッサで稼動するサーバで動作するソフトウェアである。本暗号モジュールは物理的セキュリティメカニズムを提供しない。本暗号モジュールは，製品グレードのコンポーネントである。本暗号モジュールの物理形態は，マルチチップスタンドアロン型暗号モジュールである。

６動作環境

6.1 OS

本暗号モジュールは単一オペレータ動作モードの制限下で動作させる。

本暗号モジュールを利用するユーザのアプリケーションは OS の制御下でプロセスにロードされる。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空間のメモリを参照するスレッド単位に動作する。本暗号モジュールは各スレッドから呼び出さる。アプリケーションはプロセス単位で管理され，本暗号モジュールが使用するデータ領域は呼び出したスレッドが使用するアドレス空間であり，これは OS で割り当てられた独立したプロセスのアドレス空間である。また，本暗号モジュールはプロセス間通信を使用しない。

マルチスレッドアプリケーションで本暗号モジュールを使用してもよい。本暗号モジュールはスレッド同期の機能を提供しない。

6.2 完全性

本暗号モジュールのロード時には，パワーアップ自己テストを実行し，モジュールの完全性を検査する。

承認されたアルゴリズムである RSASSA-PSS を用いて，完全性検証を行う。検証用公開鍵はモジュールに内蔵する。

７暗号鍵管理

CSP や PSP はメモリ上に平文で保持する。OS の保護機能によってそのメモリは不正なアクセスから保護される。

CSP は論理的境界内の各暗号関数実行において不要になった段階で，ゼロ化する。ゼロ化はゼロ値をメモリに代入することで実現する。

7.1 乱数ビット列生成器

本暗号モジュールのセッション開始関数(Crypto_OpenSession[7])を実行すると，システムからエントロピー入力を取得してシードを計算する。シードを内部状態の初期値とする。

本暗号モジュールの乱数生成関数を呼び出すと，内部状態から乱数を計算し，内部状態

[7] Keymta/Crypto 製品では暗号関数実行列をセッションという。各セッションは Crypto_OpenSession()関数で始まり，Crypto_CloseSession()関数で終了する。

を更新する。

乱数シードミックス関数を呼び出すと新たにエントロピー入力を取得し，このエントロピー入力データ，追加入力データと内部状態から，新たにシードを計算する。このシードを内部状態の再初期値とする。

7.2 鍵生成

乱数を用いる鍵生成関数においては，乱数生成関数を呼び出し，鍵データを生成し，出力する。鍵生成処理の末尾において，鍵データの生成に必要としたデータ領域はゼロ化する。

7.3 鍵の入出力

本暗号モジュールの物理的境界に対する CSP の入出力は，キーボード等の標準入力ポートまたはディスプレイなど標準出力ポートを通して，行われる。また，本暗号モジュールの論理的境界に対する CSP の入出力は，API を通して，プロセスが電子的に平文で行う。

7.4 鍵の格納

本暗号モジュールは，不揮発性メモリを持たず，鍵を保持しない

7.5 鍵の破棄

本暗号モジュールは，論理的境界内で不要になった CSP をゼロ化して破棄する。本暗号モジュールが論理的境界内で CSP を破棄するタイミングと対象 CSP を下記に示す。

(1) セッション終了関数(Crypto_CloseSession)を実行した場合，乱数の内部状態(シード)
(2) オブジェクト削除関数(Crypto_DeleteObject[8])を実行した場合，本関数で管理する秘密鍵(共通鍵)，プライベート鍵，他CSP

8 自己テスト

自己テストを実行する。自己テストには，パワーアップ自己テストと条件自己テストがある。

8.1 パワーアップ自己テスト

アプリケーションプロセスに本暗号モジュールをロードするときに，パワーアップ自己テストを自動的に実行する。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

[8] Keymate/Crypto 製品では鍵要素をまとめる関数を提供しており，Crypto_CreateObject()関数で，鍵要素を指定する。Crypto_DeleteObject()関数はそれを削除する関数である。まとめたものを暗号化関数や復号関数で指定する。

プ自己テストは，ソフトウェア完全性テスト，RBG エントロピーテスト，暗号アルゴリズムテストを実行する。パワーアップ自己テスト実行結果に異常が検出されたならば，パワーアップ自己テストエラーコードを返し，そのプロセスは自己テストエラー状態となる。パワーアップ自己テストエラー状態においては，状態取得関数 Crypto_GetState()のみ実行可能である。他の暗号関数は実行できない。本暗号モジュールをアンロードした後、再度ロードすればパワーアップ自己テストエラー状態から復帰できる。

(1) ソフトウェア完全性テスト

本暗号モジュールは，RSASSA-PSS を用いてモジュールの完全性テストを行う。RSASSA-PSS の公開鍵は本暗号モジュール内部に保持する。署名データは共有ライブラリまたは動的リンクライブラリとは異なる独立したファイルに保持する。共有ライブラリまたは動的リンクライブラリ，署名，公開鍵を入力して RSASSA-PSS 署名検証を行う。

(2) RBG エントロピーテスト

RBG エントロピーソースについての暗号ヘルステストを実行する。

(3) 暗号アルゴリズムテスト

本暗号モジュールで行う暗号アルゴリズムテストを表 8-1 にあげる。

表 8-1 パワーアップ自己テストにおける暗号アルゴリズムテスト

サービスのタイプ	アルゴリズム	テストの種類
公開鍵署名	ECDSA	鍵ペア整合性
	RSASSA-PKCS1-v1_5	既知解
	RSASSA-PSS	鍵ペア整合性
公開鍵守秘	RSA-OAEP	鍵ペア整合性
共通鍵暗号 128 ビットブロック暗号	AES	既知解
ハッシュ関数	SHA-256	既知解
	SHA-384	既知解
	SHA-512	既知解
メッセージ認証	HMAC	既知解
乱数ビット列生成器	Hash_DRBG	既知解

8.2 条件自己テスト

本暗号モジュールは通常動作中に，条件自己テストとして，連続乱数ビット列生成器テスト，鍵ペア整合性テスト，エントロピーテストを実行する。条件自己テストの結果は，本暗号モジュールの関数の戻り値によって本暗号モジュールから出力する。

条件自己テストの失敗時，自己テストエラーコードを返し，そのプロセスは自己テスト

エラー状態となる。自己テストエラー状態においては，CSP をゼロ化する関数であるセッション終了関数，オブジェクト削除関数，状態取得関数のみ実行可能である。他の暗号関数は実行できない。

(1) 連続乱数ビット列生成器テスト

本暗号モジュールは，連続乱数ビット列生成器テストを実装する。連続乱数ビット列生成器テストは，乱数生成ごとに実行する。

(2) 鍵ペア整合性テスト

本暗号モジュールは，表 8-2 に示す暗号アルゴリズムにおいて鍵ペア整合性テストを実装している。鍵ペア整合性テストは，鍵生成ごとに実行し，生成した鍵を検証する

表 8-2 条件自己テストにおける鍵ペア整合性テスト

サービスのタイプ	アルゴリズム
公開鍵署名	ECDSA
	RSASSA-PKCS1-v1_5
	RSASSA-PSS
公開鍵守秘	RSA-OAEP

(3) エントロピーテスト

エントロピー入力において，エントロピーソースのエラーは，条件自己テストエラーとする。

8.3 オンデマンド自己テスト

自己テスト関数実行により，オンデマンドで自己テストを実行する。ここで自己テスト内容はパワーアップ自己テストと同じである。

オンデマンド自己テストの失敗時，自己テストエラーコードを返し，そのプロセスは自己テストエラー状態となる。自己テストエラー状態においては，CSP をゼロ化する関数であるセッション終了関数，オブジェクト削除関数，状態取得関数のみ実行可能である。他の暗号関数は実行できない。

また，本暗号モジュールをアンロード後，再ロードしても第 8.1 節のパワーアップ自己テストを実行するので，オンデマンドで自己テストを開始することができる。

9 その他の攻撃への対処

セキュリティレベル 1 では，現時点において，その他の攻撃への対処要求事項はありません。

１０ 参考文献

1) IPA,「承認されたセキュリティ機能に関する仕様」, 2008/4/7.
2) IEEE Std 1363-2000, “IEEE Standard Specifications For Public-Key Cryptography”.
3) SEC 1, “Elliptic Curve Cryptography, Version 1.0, 2000/9/20”.
4) RSA Laboratories, “PKCS #1: RSA Encryption Standard”, Version 2.1, 2002/6/14.
5) NIST FIPS Publication 197 “Advanced Encryption Standard (AES)”, 2001/11/26.
6) FIPS PUB 180-2 with Change Notice 1, ″Secure Hash Standard″, 2004/2/25.
7) NIST FIPS Publication 198a, “Keyed-Hash Message Authentication Code(HMAC)”, 2002/3/6.
8) NIST SP800-90, “Recommendation for Random Number Generation Using Deterministic Random Bit Generators (Revised)”, 2007/3.
9) NIST SP800-38A, “Recommendation for Block Cipher Modes of Operation”, 2001/12.

