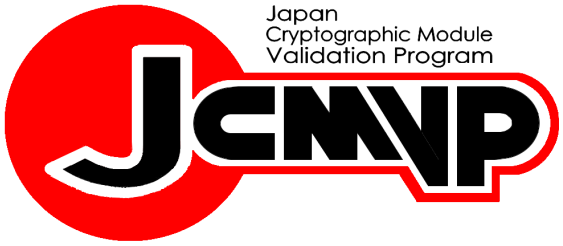


暗号アルゴリズム実装試験仕様書
– ハッシュ –

令和元年 7 月 11 日

IPA

ATR-01-C

Cryptographic Algorithm Implementation Testing Requirements

独立行政法人情報処理推進機構

目 次

1 目的

本書は, 暗号アルゴリズム実装試験ツール (JCATT) に実装されたハッシュに関する暗号アルゴリズム実装試験仕様を記述するものである. 試験の対象とする暗号アルゴリズムは, 2 章に示す通りである.

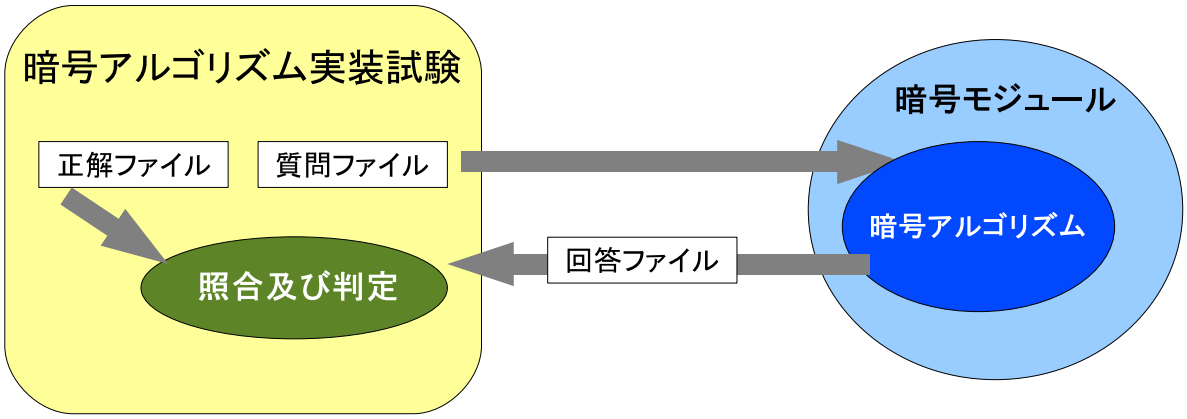
1.1 暗号アルゴリズム実装試験ツールの概要

暗号アルゴリズム実装試験ツールは次の特長を持つ.

- 試験対象の実装が暗号アルゴリズム仕様書に記述された事項に従って実装されているかどうかを試験する.
- 例えばハッシュの場合はハッシュ生成など, 各暗号が有する機能ごとに試験を行う.
- 暗号アルゴリズム実装試験ツールと試験対象の実装は, 各種ファイルを介してデータの通信を行う. このことにより, 様々なプラットフォーム上の暗号実装を試験可能となる. ここで, ツールで使う各種ファイルの内訳は以下のとおりである.
  - 質問ファイル: 暗号アルゴリズム実装試験ツールが生成するファイル. 暗号アルゴリズムに対する入力データ及び制御情報が記録されている. 暗号モジュール試験機関からベンダ側へ送る.
  - 正解ファイル: 暗号アルゴリズム実装試験ツールが質問ファイルと同時に生成するファイル. 暗号アルゴリズムに対する入力データ, 制御情報及び対応する出力データが記録されている. 暗号モジュール試験機関で保存し, 回答ファイルが送られてきた際に回答ファイルと照合する.
  - 回答ファイル: ベンダ側で, 質問ファイルを元に暗号モジュールが生成したテキストファイル. ベンダから暗号モジュール試験機関側へ送る.

ファイルフォーマットは文献 [3], サンプルファイルは文献 [4] を参照.
暗号アルゴリズム実装試験の流れは, 図 1.1 の通りである.

図 1.1: 暗号アルゴリズム実装試験の流れ



1.2 本書の構成

本書の以降の構成は次の通りである.

- 2 章: 暗号アルゴリズム実装試験ツールが試験の対象とする暗号アルゴリズムを示す.
- 3 章: 各暗号の試験項目を記述する.
- 4 章: bit 列と byte 列との間の変換規則を記述する.

なお, 本書を通して次の略語を使用する.

- JCATT: 暗号アルゴリズム実装試験ツール
- IUT: JCATT が試験の対象とする実装

2　本書で対象とする承認されたセキュリティ機能

本書が対象とする暗号アルゴリズムを次に示す.

2.1　FIPS 180-4 に記載されたハッシュ関数

- SHA-1
- SHA-224
- SHA-256
- SHA-384
- SHA-512
- SHA-512/224
- SHA-512/256

2.2　FIPS 202 に記載されたハッシュ関数

- SHA3-256
- SHA3-384
- SHA3-512

2.3　FIPS 202 に記載された可変長出力関数 (XOF: extendable-output function)

- SHAKE128
- SHAKE256

3 暗号アルゴリズム実装試験仕様 – ハッシュ関数 –

3.1 FIPS 180-4 に記載されたハッシュ関数

ハッシュ関数アルゴリズム, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256 の暗号アルゴリズム実装試験項目を記述する.

また, 本ツールの試験対象である上記 7 つのハッシュ関数は, ハッシュ値のサイズを除いて入出力は同じであるので, 全てのハッシュ関数機能に対して試験項目は同じである. 以下に試験項目を記述する.

- 短いメッセージに対する試験 (SMT)
- 選択された長いメッセージに対する試験 (SLMT)
- 擬似ランダムメッセージに対する試験 (PGMT)

ハッシュ関数モジュールが byte oriented 実装の場合の上記試験項目の詳細を以下に記述する. 試験項目の詳細は SHAVS[1] に従っている. ただし, SLMT で行う試験におけるビット長は別に定める規定値とする.

3.1.1 短いメッセージに対する試験 (SMT)

ハッシュ関数のブロック長 (ビット数) を m とする. すなわち,

- SHA-1: $m = 512$
- SHA-224: $m = 512$
- SHA-256: $m = 512$
- SHA-384: $m = 1,024$
- SHA-512: $m = 1,024$
- SHA-512/224: $m = 1,024$
- SHA-512/256: $m = 1,024$

となる. この試験項目では, $m/8+1$ 個のランダムに生成されたメッセージに対するハッシュ値に対する Known Answer Test を行う. 各メッセージのビット長は $0, 8, 16, \ldots, m$ である.

3.1.2 選択された長いメッセージに対する試験 (SLMT)

3.1.1 節のように, ハッシュ関数のブロック長 (ビット数) を m とする. この試験項目では, ランダムに生成された $m/8$ 個の長いメッセージに対するハッシュ値に対する Known Answer Test を行う. 各メッセージのビット長は, $m+8\times i\times$ (“Upperbound of SLMT” -1), $1 \le i \le m/8$, である. Upperbound of SLMT は別途定める規定値とする.

3.1.3 擬似ランダムメッセージに対する試験 (PGMT)

与えられた Seed, outerloop および innerloop から, 次のアルゴリズムにより計算されるデータ MD[0]～MD[outerloop-1] に対する Known Answer Test を行う. (注: 実際のプログラムでは配列 MD[ ] は outerloop 個の大きさを確保する必要はない)

```
for (j=0; j<outerloop; j++)
{
  MD[0] = Seed;
  MD[1] = Seed;
  MD[2] = Seed;
```

```
  for (i=3; i<innerloop+3; i++)
  {
    M[i] = MD[i-3]||MD[i-2]||MD[i-1]; // 記号||はデータの連結
    MD[i] = Hash(M[i]); // ハッシュ関数
  }
  MD[j] = MD[i-1];
  Seed  = MD[i-1];
  OUTPUT MD[j];
}
```

3.2 FIPS 202 に記載されたハッシュ関数

ハッシュ関数アルゴリズム, SHA3-256, SHA3-384, SHA3-512 の暗号アルゴリズム実装試験項目を記述する.

また, 本ツールの試験対象である上記 4 つのハッシュ関数は, 全てのハッシュ関数機能に対して試験項目は同じである. 以下に試験項目を記述する.

- 短いメッセージに対する試験 (SMT)
- 選択された長いメッセージに対する試験 (SLMT)
- 擬似ランダムメッセージに対する試験 (PGMT)

試験項目の詳細は SHA3VS[2] に従っている.

3.2.1 短いメッセージに対する試験 (SMT)

ハッシュ関数のブロック長 (ビット数) を r とする. すなわち,

- SHA3-256: $r = 1,088$
- SHA3-384: $r = 832$
- SHA3-512: $r = 576$

となる.

3.2.1.1 bit oriented 実装に対する短いメッセージに対する試験 (SMT)

この試験項目では, $r+1$ 個のランダムに生成されたメッセージに対するハッシュ値に対する Known Answer Test を行う. 各メッセージのビット長は $0,1,2,\ldots,r$ である. bit 列と byte 列との変換は, 4 章を参照のこと.

3.2.1.2 byte oriented 実装に対する短いメッセージに対する試験 (SMT)

この試験項目では, $r/8+1$ 個のランダムに生成されたメッセージに対するハッシュ値に対する Known Answer Test を行う. 各メッセージのビット長は $0,8,16,\ldots,r$ である.

3.2.2 選択された長いメッセージに対する試験 (SLMT)

この試験項目では, ランダムに生成された “Upperbound of SLMT” 個の長いメッセージに対するハッシュ値に対する Known Answer Test を行う. Upperbound of SLMT は別途定める規定値とする. 3.2.1 節のように, ハッシュ関数のブロック長 (ビット数) を r とする.

3.2.2.1 bit oriented 実装に対する選択された長いメッセージに対する試験 (SLMT)

各メッセージのビット長は, $r+i\times(r+1)$, $1 \leq i \leq$ “Upperbound of SLMT”, である.

3.2.2.2 byte oriented 実装に対する選択された長いメッセージに対する試験 (SLMT)

各メッセージのビット長は, $r+i\times(r+8)$, $1 \leq i \leq$ “Upperbound of SLMT”, である.

3.2.3 擬似ランダムメッセージに対する試験 (PGMT)

与えられた *Seed*, *outerloop* および *innerloop* から, 次のアルゴリズムにより計算されるデータ (*outerloop* 個の MD_0) に対する Known Answer Test を行う.

記号の定義

擬似コード上の表記	型	説明
i	整数	1 から *innerloop* までの値をとる整数.
innerloop	整数	内側ループ回数
j	整数	0 から *outerloop* − 1 までの値をとる整数.
MD_i	ビット列	主として SHA-3 関数を呼び出すことにより生成されたビット列を格納する.
msg_i	ビット列	SHA-3 関数への入力メッセージ.
outerloop	整数	外側ループ回数
Seed	ビット列	初期入力メッセージ.

注
1. 網掛け ( ) された項目については, 質問ファイルで提供される.

関数定義

擬似コード上の表記	説明
Hash(M)	SHA3-256, SHA3-384 又は SHA3-512. 入力パラメータとして, メッセージ M をとる.
Output(V)	V を回答ファイルに出力する.

擬似ランダムメッセージに対する試験 (PGMT) のアルゴリズム

Algorithm 1 PGMT Algorithm for SHA-3

1: $MD_0 = Seed$
2: **for** $j = 0$ to $outerloop - 1$ **do**
3: **for** $i = 1$ to *innerloop* **do**
4: $msg_i = MD_{i-1}$
5: $MD_i =$**Hash**(msg_i) ▷ SHA-3 関数を呼び出す
6: **end for**
7: $MD_0 = MD_{innerloop}$
8: **Output**(MD_0) ▷ 生成されたビット列を回答ファイルに記録する
9: **end for**

3.3 FIPS 202 に記載された可変長出力関数 (XOF)

可変長出力関数, SHAKE128, SHAKE256 の暗号アルゴリズム実装試験項目を記述する.

また, 本ツールの試験対象である上記 2 つの可変長出力関数は, 全ての可変長出力関数に対して試験項目は同じである. 以下に試験項目を記述する.

- 短いメッセージに対する試験 (SMT)
- 選択された長いメッセージに対する試験 (SLMT)
- 擬似ランダムメッセージに対する試験 (PGMT)
- 可変長出力試験 (VOT)

試験項目の詳細は SHA3VS[2] に従っている.

3.3.1 短いメッセージに対する試験 (SMT)

可変長出力関数のブロック長 (ビット数) を r とする. すなわち,

- SHAKE128: $r = 1,344$
- SHAKE256: $r = 1,088$

となる.

3.3.1.1 bit oriented 実装に対する短いメッセージに対する試験 (SMT)

この試験項目では, $2 \times r + 1$ 個のランダムに生成されたメッセージに対する可変長出力関数に対する Known Answer Test を行う. 各メッセージのビット長は $0, 1, 2, \ldots, (2 \times r)$ である. ただし, SMT で行う試験における出力ビット長は別に定める規定値とする. bit 列と byte 列との変換は, 4 章を参照のこと.

3.3.1.2 byte oriented 実装に対する短いメッセージに対する試験 (SMT)

この試験項目では, $r/4 + 1$ 個のランダムに生成されたメッセージに対する可変長出力関数に対する Known Answer Test を行う. 各メッセージのビット長は $0, 8, 16, \ldots, r/4$ である. ただし, SMT で行う試験における出力ビット長は別に定める規定値とする.

3.3.2 選択された長いメッセージに対する試験 (SLMT)

この試験項目では, ランダムに生成された “Upperbound of SLMT” 個の長いメッセージに対する可変長出力関数に対する Known Answer Test を行う. Upperbound of SLMT は別途定める規定値とする. 3.3.1 節のように, 可変長出力関数のブロック長 (ビット数) を r とする. ただし, SLMT で行う試験における出力ビット長は別に定める規定値とする.

3.3.2.1 bit oriented 実装に対する選択された長いメッセージに対する試験 (SLMT)

各メッセージのビット長は, $r + i \times (r + 1)$, $1 \leq i \leq$ “Upperbound of SLMT”, である.

3.3.2.2 byte oriented 実装に対する選択された長いメッセージに対する試験 (SLMT)

各メッセージのビット長は, $r + i \times (r + 8)$, $1 \leq i \leq$ “Upperbound of SLMT”, である.

3.3.3 擬似ランダムメッセージに対する試験 (PGMT)

この試験項目では, 与えられた $minoutlen, maxoutlen, msg, outerloop$ 及び $innerloop$ から, 次のアルゴリズムにより計算されるデータ { ($outputlen_0$, $output_0$), ..., ($outputlen_{outerloop-1}$, $output_{outerloop-1}$) } に対する Known Answer Test を行う.

記号の定義

擬似コード上の表記	型	説明
i	整数	1 から $innerloop$ までの値をとる整数.
$innerloop$	整数	内側ループ回数
j	整数	0 から $outerloop-1$ までの値をとる整数.
$maxoutlen$	整数	ビット数で指定する.
$maxoutbytes$	整数	$\lfloor maxoutlen/8 \rfloor$.
$minoutlen$	整数	ビット数で指定する.
$minoutbytes$	整数	$\lceil minoutlen/8 \rceil$.
msg	ビット列	初期入力メッセージ.
msg_i	ビット列	SHAKE 関数への入力メッセージ.
$outerloop$	整数	外側ループ回数
$output_{j,i}$	ビット列	SHAKE 関数を呼び出すことにより生成されたビット列の組の (j,i) 番目の要素.
$output_j$	ビット列	回答ファイルに出力されるビット列. $output_{j,innerloop}$ に等しい.
$outputlen$	整数	SHAKE 関数の呼出に使用する変数で, SHAKE 関数の出力ビット長を規定する.
$outputlen_j$	整数	SHAKE 関数の $(j, innerloop)$ 番目の呼出に使用された変数で, $output_j$ のビット長を表す.
$range$	整数	SHAKE 関数の出力バイト長の変動幅を表す変数.
$rightmost\_output\_bits$	整数	SHAKE 関数の出力の右から 16 ビットの整数表現.

注
1. 網掛け ( ) された項目については, 質問ファイルで提供される.

関数定義

擬似コード上の表記	説明
leftmost(V,a)	ビット列 V の左から a ビット
len(V)	ビット列 V のビット長
$x \mod n$	n を法とした x の剰余
Output(a,V)	a 及び V を回答ファイルに出力する.
rightmost(V,a)	ビット列 V の右から a ビット
SHAKE(M,a)	SHAKE128 又は SHAKE256. 入力パラメータとして, メッセージ M, 出力長 a をとる.
$U \| \| V$	ビット列 U と V とを連結したビット列

Algorithm 2 PGMT Algorithm for XOF

$outputlen = 8 * \lfloor maxoutlen/8 \rfloor$
$output_0 = msg$
for $j = 0$ to $outerloop - 1$ **do**
    for $i = 1$ to $innerloop$ **do**
        if **len**$(output_{i-1}) \geq 128$ **then**
            $msg =$ **leftmost**$(output_{i-1}, 128)$ ▷ 入力メッセージ用に 128 ビットを切り出す
        else
            $msg = output_{i-1} || 0^{128-\textbf{len}(output_{i-1})}$
        end if
        $output_{j,i} =$**SHAKE**$(msg, outputlen)$ ▷ SHAKE 関数を呼び出す
        if $i = innerloop$ **then**
            $outputlen_j = outputlen$ ▷ 回答ファイルに記録するための変数に保存する
        end if
        $rightmost\_output\_bits =$ **rightmost**$(output_{j,i}, 16)$ ▷ 右から 16 ビットを切り出す
        $range = (maxoutbytes - minoutbytes + 1)$ ▷ 出力バイト長の変動幅を計算する
        $outputlen = minoutbytes + (rightmost\_output\_bits \mod range)$
            ▷ 次回 SHAKE 関数呼び出し時の出力ビット長を計算しておく
    end for
    $output_j = output_{j,innerloop}$
    Output$(outputlen_j, output_j)$ ▷ 生成されたビット列を回答ファイルに記録する
end for

擬似ランダムメッセージに対する試験 **(PGMT)** のアルゴリズム

3.3.4 可変長出力試験 (VOT)

この試験項目では, あるビット長 *input_len* のメッセージを入力し, IUT が許容する出力ビット長の範囲 (*minoutlen* から *maxoutlen* まで) で, 出力ビット長を変化させ、出力された可変長出力に対する Known Answer Test を行う. 可変長出力試験における出力ビット長の初期値は *minoutlen* である. bit 列と byte 列との変換は, 4 章を参照のこと.

4 bit 列と byte 列との変換

4.1 bit 列から byte 列への変換

Algorithm 3 bitstring to a byte string

Input: ビット長 $sLen$ のビット列 S

Output: バイト長 $\lceil sLen/8 \rceil$ のバイト列 X

1: $m = \lceil sLen/8 \rceil$ とおく.

2: $T = S||0^{-sLen \mod 8}$ とおく.

3: $0 \leq i < m, 0 \leq j \leq 7$ なる整数の組 (i, j) に対して, T の左から $(8i+j)$ 番目のビットを $b_{i,j}$ とおく.

4: **for** $i = 0$ to $m-1$ **do**

5: $\quad x_i = \sum_{j=0}^{7} b_{i,j} \cdot 2^j.$

6: **end for**

7: $X = x_0 x_1 x_2 \ldots x_{m-1}$ を出力する

4.2 byte 列から bit 列への変換

Algorithm 4 byte string to a bitstring

Input: バイト長 n のバイト列 X, 変換後のビット列 S のビット長 $sLen$, 但し $sLen$ は, $8(n-1) < sLen \leq 8n$ を満たす.

Output: ビット列 S

1: バイト列 X の, 左から i 番目 (但し, $i \in \{0, \ldots, n-1\}$ とする) のバイトを x_i とおく.

2: $0 \leq i < n, 0 \leq j \leq 7$ なる整数の組 (i, j) に対して次を満たす $b_{i,j} \in \{0, 1\}$ を決定する.

$x_i = \sum_{j=0}^{7} b_{i,j} \cdot 2^j$

3: $S = b_{0,0} b_{0,1} b_{0,2} b_{0,3} b_{0,4} b_{0,5} b_{0,6} b_{0,7} \ldots b_{n-1,0} b_{n-1,1} \ldots b_{n-1,(sLen-8n+7)}$ を出力する

5 確認書発行条件

5.1 パラメータについて

5.1.1 FIPS 180-4 に記載されたハッシュ関数

ハッシュ関数において, 暗号アルゴリズム確認書を発行するための条件は, 網掛けされた試験対象機能を少なくとも 1 個実装し, 暗号アルゴリズム実装試験に合格することである. 暗号アルゴリズム実装試験に使用するパラメータの入力条件及びその既定値は, 表 5.1 に記載する値とする.

表 5.1: ハッシュ関数の既定値及び入力条件

試験対象機能	入力欄		既定値	入力条件
ハッシュ関数	SLMT	Upperbound of SLMT	100	100 以上
	PGMT	内側ループ回数	1000	1000 以上
		外側ループ回数	100	100 以上

5.1.2 FIPS 202 に記載されたハッシュ関数

ハッシュ関数において, 暗号アルゴリズム確認書を発行するための条件は, 網掛けされた試験対象機能を少なくとも 1 個実装し, 暗号アルゴリズム実装試験に合格することである. 暗号アルゴリズム実装試験に使用するパラメータの入力条件及びその既定値は, 表 5.2 に記載する値とする.

表 5.2: ハッシュ関数の既定値及び入力条件

試験対象機能	入力欄		既定値	入力条件
ハッシュ関数	SLMT	Upperbound of SLMT	100	100 以上
	PGMT	内側ループ回数 *innerloop*	1000	1000 以上
		外側ループ回数 *outerloop*	100	100 以上

5.1.3 FIPS 202 に記載された可変長出力関数

可変長出力関数において, 暗号アルゴリズム確認書を発行するための条件は, 網掛けされた試験対象機能を少なくとも 1 個実装し, 暗号アルゴリズム実装試験に合格することである. 暗号アルゴリズム実装試験に使用するパラメータの入力条件及びその既定値は, 表 5.3 に記載する値とする.

表 5.3: 可変長出力関数の既定値及び入力条件

試験対象機能	入力欄		既定値	入力条件
	SMT	出力ビット長	128	SHAKE128 を選択時, 次により定まる • 128 又は実装がサポートする出力ビット長の最大値のいずれか小さい方 SHAKE256 を選択時, 次により定まる • 256 又は実装がサポートする出力ビット長の最大値のいずれか小さい方
	SLMT	出力ビット長	128	SHAKE128 を選択時, 次により定まる • 128 又は実装がサポートする出力ビット長の最大値のいずれか小さい方 SHAKE256 を選択時, 次により定まる • 256 又は実装がサポートする出力ビット長の最大値のいずれか小さい方
		Upperbound of SLMT	100	100 以上
	PGMT	IUT が許容する出力ビット長の最小値 *minoutlen*	16	• 8 の倍数 • 16 以上 • *maxoutlen* 以下
		IUT が許容する出力ビット長の最大値 *maxoutlen*	65536	• 8 の倍数 • *minoutlen* 以上 • 65536 以下
		内側ループ回数 *innerloop*	1000	1000 以上
		外側ループ回数 *outerloop*	100	100 以上
	VOT	入力メッセージのビット長 *input_len*	• SHAKE128 の場合, 128 • SHAKE256 の場合, 256	
		IUT が許容する出力ビット長の最小値 *minoutlen*	16	• 8 の倍数 • 16 以上 • *maxoutlen* 以下
		IUT が許容する出力ビット長の最大値 *maxoutlen*	65536	• 8 の倍数 • *minoutlen* 以上 • 65536 以下

附 則
この手順は, 平成 21 年 1 月 23 日から施行し, 平成 21 年 1 月 8 日から適用する.

附 則
この手順は, 平成 21 年 7 月 1 日から施行し, 平成 21 年 7 月 10 日から適用する.

附 則
この手順は, 平成 24 年 2 月 29 日から施行し, 平成 24 年 6 月 1 日から適用する.

附 則
この手順は, 平成 30 年 6 月 22 日から施行し, 平成 30 年 6 月 22 日から適用する.

附 則
この手順は, 令和元年 7 月 11 日から施行し, 令和元年 7 月 11 日から適用する.

参考文献

[1] Lawrence E. Bassham III, and Timothy A. Hall, *The secure hash algorithm validation system (SHAVS)*, National Institute of Standards and Technology, July 23, 2012.

[2] Sharon S. Keller, and Lawrence E. Bassham III, *The Secure Hash Algorith 3 Validation System (SHA3VS)*, National Institute of Standards and Technology, April 7, 2016.

[3] JCATT ファイルフォーマット仕様書 – ハッシュ –, `https://www.ipa.go.jp/security/jcmvp/documents/open/jcatt/format/jcatt_fileformat_c.zip`

[4] JCATT サンプルファイル – ハッシュ –, `https://www.ipa.go.jp/security/jcmvp/documents/open/jcatt/sample/jcatt_sample_c.zip`

改版履歴

識別番号	ATR-01-C

改訂年月日	作成者・承認者	改訂内容
平成 21 年 1 月 23 日	橋本・仲田	新規制定
平成 30 年 6 月 22 日	櫻井・江口	一部改正 (承認されたセキュリティ機能の改正に伴い, RIPEMD-160 の記述を削除し, SHA-512/224, SHA-512/256 の記述を追加. また, SHA-3, SHAKE に対する 試験要件を追加.)
令和元年 7 月 11 日	櫻井・江口	一部改正 (誤植を訂正)