ESIGN 認証仕様書

1　まえがき

「ESIGN 認証」は、高い安全性をもったディジタル署名「ESIGN 署名」に基づき作られたものである。ESIGN 署名は、基本 ESIGN 署名関数（基本暗号プリミティブ）[10] に基づき高い安全性をもつ暗号スキームに変換されたものである [6]。

なお、これら変換法では、補助関数としてハッシュ関数を利用するが、ハッシュ関数としては、出力サイズの条件の合致した任意の衝突困難な一方向性ハッシュ関数を用いることができるため、特定のハッシュ関数に限定する必要はない。本資料では、その一実現例として、SHA-1 に基づく方式 [2] を例示する（9章参照）。なお、同時に提出するテストデータ、サンプルプログラムではここで例示したハッシュ関数を用いて作られている。

補助アルゴリズムとしては、上で述べたハッシュ関数以外に、素数生成アルゴリズムなどを必要とするが、それについては、refhojo 章を参照されたい。

2　設計方針、設計基準

2.1　基本方針

実用的暗号において（むしろ実用的であればあるほど）様々な実用的な利用環境での安全性を理論的に保証することは重要である。例えば、1999 年に起こった RSA に基づくメッセージ復元型署名（ISO 9796）に対する署名偽造攻撃の成功などにより、ISO の標準化の対象となるような実用的なディジタル署名方式においても、理論的な安全性の証明の重要性が広く認識されるようになった。つまり、現在知られている署名攻撃の分類で最強の意味の安全性を持つ（つまり、適応的選択文書攻撃に対して存在的偽造不可である）[7] ことが（適当な仮定のもとで）証明できるような実用的な署名方式が求められるようになった。

利用者認証においても同様で、最強の意味での（攻撃者の能動的な攻撃を許した下で、利用者のなりすましが不可であるという意味での）安全性が（適当な仮定のもとで）証明できるような実用的な利用者認証方式が求められるようになった

ESIGN 認証はそのような要求条件に答えるために作られた利用者認証目的の公開鍵暗号方式である。そのような最強の意味の安全性を持った実用的な認証を設計するために我々がとった方針は、（適当な仮定のもとで）最強の意味の安全性を持った（適応的選択文書

攻撃に対して存在的偽造不可である）ディジタル署名であるESIGN 署名を用いて利用者認証を構成することである。最強の意味の安全性を持ったディジタル署名を用いれば、最強の意味の安全性を持った認証を設計することは極めて簡単である。

ゼロ知識証明（例えば、Fiat-Shamir 法）や3 交信プロトコル（例えば、Shnorr 法）などに比べて、このアプローチが優れている点は、その交信回数の少なさである。ESIGN 認証は、認証者と被認証者の交信回数が2 回である。それに比べて、ゼロ知識証明では、交信回数が4 回以上、3 交信プロトコルでは3 回必要である。

さらに、ESIGN 認証は、高い安全性のみならず、極めて高性能な ESIGN 署名に基づき構成されているため、他のいずれの認証方式よりも高い実用性を保持している。

2.2 ESIGN 署名

ESIGN 署名はハッシュ関数を用いて基本的署名関数 [10] を最強の意味の安全性を持つような署名に変換した方式である。ここで用いられたハッシュ関数が理想的にランダムであると仮定すると（基本署名関数の安全性の仮定に加えて）、最強の意味の安全性を持つことが証明できる。このようなアプローチは、1994 年のBellare, Rogaway の OAEP や PSS 署名の提案以来、標準的なアプローチとされているものである。

実用的なハッシュ関数が理想的なランダム関数という仮定は強い仮定であるが、もしこのような仮定の下で安全性の証明のついた方式に対して何らかの攻撃があれば、それはそのハッシュ関数にランダム関数に無い性質（非ランダム性）の一つが見つかったことを意味する。SHA のように注意深く作られたハッシュ関数の場合、最も基本的な非ランダム性である衝突性ですら見つけることが困難であると信じられている。従って、ランダムオラクルモデルの下で安全性の証明のついた方式を破ることは非常に困難であると信じられている。これが、ランダムオラクルモデルにおける安全性のパラダイムである[1]。

安全性の観点では、ESIGN 署名は、RSA 仮定[2]の近似版である e 乗恨近似仮定（基本ESIGN 署名関数の一方向性仮定）とランダムオラクルモデルの下で最強の安全性をもつことが証明できる [6]。

---

[1] Canetti らは、ランダムオラクルモデルで証明ができてもどのような計算量的仮定の下でもそのモデルを実現するハッシュ関数が構成できないような暗号プロトコルを示している [4] 。しかし、この例は極めて人工的に作られたものであり、自然に構成された方式に対しては、Canetti たちの結果に基づく懸念は特に意味がないように思われる。

[2] RSA 関数が一方向であるという仮定。つまり、RSA 暗号の公開鍵 (e, n) および $y \in \boldsymbol{Z}/n\boldsymbol{Z}$ が与えられたとき、$y = x^k \bmod n$ となるような $x \in \boldsymbol{Z}/n\boldsymbol{Z}$ を見つけることが困難である（多項式時間アルゴリズムでの成功確率が無視できる程度である）ことを RSA 仮定と呼ぶ。

効率性の観点では、ESIGN 署名は、RSA に基づくいずれの署名に比べても署名生成処理が数 10 倍程度高速であり、署名検証処理はほぼ同等である。楕円曲線に基づく署名方式と比べても署名生成処理、署名検証処理のいずれでも数倍高速である。

3 記法

本仕様書では、以下のような表記を用いる。

- $a := b$: b の値を a に代入する。もしくは、a を b として定義する。
- $\mathbf{Z}$: 整数の集合。
- $\boldsymbol{Z}/n\boldsymbol{Z} := \{0, 1, \ldots, n-1\}$。
- A, B を集合とするとき、$A \backslash B := \{x \mid x \in A \wedge x \notin B\}$。
- $(\boldsymbol{Z}/n\boldsymbol{Z})^* := \{1, 2, \ldots, n-1\} \backslash \{x \mid \gcd(x, n) \neq 1\}$。
- $\{0,1\}^*$ は、有限長のビット列の集合。$\{0,1\}^*$ を $\mathbf{B}$ と記すこともある。
- $\{0,1\}^i$ は、i ビット長のビット列の集合。$\{0,1\}^i$ を $\mathbf{B}_i$ と記すこともある。
- $a \in \mathbf{Z}$ のとき、$\mathbf{B}_i[a]$ は、

  $$a = a_0 + 2a_1 + 2^2 a_2 + \cdots 2^{i-1} a_{i-1}$$

  となるようなビット列 $(a_{i-1}, a_{i-2}, \ldots, a_0) \in \mathbf{B}_i$ を意味する。
- $a := (a_{i-1}, a_{i-2}, \ldots, a_0) \in \mathbf{B}_i$ のとき、$\mathbf{I}[a]$ は、

  $$b = a_0 + 2a_1 + 2^2 a_2 + \cdots 2^{i-1} a_{i-1}$$

  となるような整数 $b \in \mathbf{Z}$ を意味する。
- $a \in \mathbf{B}_i$ のとき、$|a| := i$.
- $a \equiv b \pmod{n}$ は、$a - b$ が p で割り切れることを意味する。$a := b \bmod n$ は、$a \in \boldsymbol{Z}/n\boldsymbol{Z}$ かつ $a \equiv b \pmod{n}$ を意味する。
- $a \in \mathbf{B}$ かつ $b \in \mathbf{B}$ のとき、$a||b$ は a と b の結合を意味する。例えば、$(0,1,0,0)||(1,1,0) = (0,1,0,0,1,1,0)$ となる。

- $X \in \mathbf{B}$ のとき、$[X]^{mLen}$ は、X の最上位 $mLen$ ビットを意味する。例えば、$X = (0,1,1,0,0,1,0,0,0,1)$ のとき、$[X]^4 = ((0,1,1,0)$ である。
- $a \in \mathbf{B}_i$ かつ $b \in \mathbf{B}_i$ のとき、$a \oplus b$ はビット毎の排他的論理和を意味する。（つまり、$a \oplus b \in \mathbf{B}_i$ となる。）

4 基本署名関数（暗号プリミティブ）

ESIGN 署名は、基本暗号関数（暗号プリミティブ）として、基本 ESIGN 署名関数 [10] を用いている。以下に、基本 ESIGN 署名関数（鍵生成 $\mathcal{G}$、署名生成 §、署名検証 $\mathcal{V}$）を示す。

4.1 鍵生成: $\mathcal{G}$

$\mathcal{G}$ の入力と出力は以下のとおりである。

[入力] セキュリティパラメータ $k(= pLen) \in \mathbf{Z}$ （正整数）。

[出力] 公開鍵, $(n, e, pLen) \in \mathbf{Z}^3$ と秘密鍵 $(p, q) \in \mathbf{Z}^2$ の対。

入力 k における $\mathcal{G}$ の処理は以下の通りである。

- 2 つの素数 p, q $(2^{k-1} \leq p, q \leq 2^k - 1)$ を生成し、$n := p^2 q$ を計算する。
- 整数 $e > 4$ (例えば、$e = 2^l, l = O(\log n)$) を選択する。
- $pLen = k$ を定める。

4.2 署名生成: $\mathcal{S}$

$\mathcal{S}$ の入力と出力は以下の通りである。

[入力] メッセージ $m \in \{0,1\}^{pLen-1}$ および公開鍵 $(n, e, pLen) \in \mathbf{Z}^3$ と秘密鍵 $(p, q) \in \mathbf{Z}^2$。

[出力] 署名 $s \in \mathbf{Z}$.

$\mathcal{S}$ の処理は以下の通りである。

1. ランダムかつ一様に $(\boldsymbol{Z}/pq\boldsymbol{Z})\backslash p\mathbf{Z} := \{r \in \boldsymbol{Z}/pq\boldsymbol{Z} | \gcd(r, p) = 1\}$ から r を選ぶ。

2. $z := (0||m||0^{2 \cdot pLen}) \in \{0,1\}^{3 \cdot pLen}$ と $\alpha := (\mathbf{I}[z] - r^e) \bmod n$ を求める。

3. 以下のようにして (w_0, w_1) を求める。

$$w_0 := \lceil \frac{\alpha}{pq} \rceil, \tag{1}$$

$$w_1 := w_0 \cdot pq - \alpha. \tag{2}$$

もし $w_1 \geq 2^{2 \cdot pLen - 1}$ であれば、Step 1 にもどる。(つまり, もし $\mathbf{B}_{2 \cdot pLen}[w_1]$ の最上位ビットが 1 の場合、Step 1 にもどる。)

4. $t := \frac{w_0}{er^{e-1}} \bmod p$, および $s := (r + tpq) \bmod n$ を計算する。

5. m の署名として s を出力する。

4.3 署名検証: $\mathcal{V}$

$\mathcal{V}$ の入力と出力は以下の通りである。

[入力] 署名 $s \in \mathbf{Z}$、文書 $m \in \{0,1\}^{pLen-1}$ および公開鍵 $(n, e, pLen) \in \mathbf{Z}^3$。

[出力] 検証結果 (「正当」もしくは「不正」)。

$\mathcal{V}$ の処理 は、以下の通りである。

- $$b := \mathbf{B}_{3 \cdot pLen}[s^e \bmod n]$$

  としたとき、以下の式が満足されるかどうかをチェックする：

$$[b]^{pLen} = 0||m. \tag{3}$$

- もし満足されれば,「正当」(もしくは 1) を出力する。さもなければ、「不正」(もしくは 0) を出力する。

5 補助関数

以下に、本仕様において使用される補助関数を示す。

- (k ビット) 乱数生成アルゴリズムもしくは (入力された整数 k に対して) k ビットの疑似乱数を生成出力するアルゴリズム：

本仕様においては、鍵生成ならびに暗号化処理において、ある範囲からランダムな値を選択する必要があり、そのとき乱数もしくは疑似乱数を利用する。物理的に発生された乱数源を用いる場合には、そのための補助装置が必要となる。そのような補助装置を用いない場合には、疑似乱数を利用する。疑似乱数生成アルゴリズムの例は、IEEE P1363 [8] の Annex D.6 もしくは、[9] の 6 章に示されているようなアルゴリズムを利用することができる。

- (k ビット) 素数生成アルゴリズム：

  (入力された整数 k に対して) k ビットの素数を出力するアルゴリズム。例として、(k ビットの) 乱数を上記、乱数生成アルゴリズムにより生成し、Miller-Rabin の素数判定テストを t 回繰り返してパスするまで、乱数の生成を繰り返す。入力された乱数が Miller-Rabin テストを t 回繰り返してパスした場合、その乱数を素数と判定し出力するアルゴリズム。ここで、Miller-Rabin の素数判定テストは、IEEE P1363 [8] Annex A.15.1 の Miller-Rabin の素数判定テストの仕様に従うものとする。なお、$k(\geq 88)$ ビットの整数が、上記の Miller-Rabin の素数判定テストを $t(> 1)$ 回パスしたにもかかわらず、合成数である確率は、高々

$$p_{k,t} = 2^{t+4}k(2^{-\sqrt{tk}})\sqrt{\frac{k}{t}}$$

  であることが知られている [5]。

- ハッシュ関数：

  その典型的な構成例を 9 章で示す。そこに示されるハッシュ関数の構成例においては、SHA-1 を用いる。これは、NIST によって提案された、Secure Hash Algorithm (SHA-1) を指す。SHA-1 の仕様は、FIPS 180-1 standard [11] に従うものとする。

- 基本的な整数演算アルゴリズム：

  IEEE P1363 [8] の Annex A.1 – A.3 などに示されているアルゴリズムに従うものとする。

6 ESIGN 認証仕様

6.1 準備

この章では、利用者認証方式 ESIGN 認証について述べる。

6.2 手順

利用者 A が、検証者 B に正当な利用者であることを証明するプロトコルを示す。

[事前処理]

最初に事前手続きとして、利用者 A は， ESIGN 署名の鍵生成演算 $\mathcal{G}$ を用いて、公開鍵 $(n, e, HID, pLen)$ と秘密鍵 (p, q) の対を作成する。 A は、秘密鍵を秘密に保持すると同時に、公開鍵 $(n, e, HID, pLen)$ を検証者 B に登録する。

[認証処理]

ステップ 1： 検証者 B は乱数 $r \in \{0,1\}^{mLen}$ を生成して，それを利用者に送る.

ステップ 2： 利用者 A は，自分だけが知る秘密鍵 (p, q) を用いて ESIGN 署名演算 $\mathcal{S}$ を用いて、 r に対する署名 s を計算し，それを検証者 B に送る.

ステップ 3： 検証者 B は，利用者 A の公開鍵 $(n, e, HID, pLen)$ を用いて r に対する署名 s の正当性を ESIGN 署名検証演算 $\mathcal{V}$ により検証する。つまり、以下の式が満足されるかどうかをチェックする：

$$[\mathbf{B}_{3 \cdot pLen}[s^e \bmod n]]^{pLen} = 0 || H(r). \tag{4}$$

7 ESIGN 署名

7.1 準備

この章では、デジタル署名方式 ESIGN について述べる。これは、三つ組 $(\mathcal{G}, \mathcal{S}, \mathcal{V})$ によって定められる。 $\mathcal{G}$ は鍵生成演算, $\mathcal{S}$ は署名演算, $\mathcal{V}$ は検証演算である。

7.2 鍵生成: $\mathcal{G}$

$\mathcal{G}$ の入力と出力は以下のとおりである。

[入力] セキュリティパラメータ $k(= pLen) \in \mathbf{Z}$ （正整数）。

[出力] 公開鍵, $(n, e, HID, pLen) \in \mathbf{Z}^3$ と秘密鍵 $(p, q) \in \mathbf{Z}^2$ の対。

入力 k における $\mathcal{G}$ の処理は以下の通りである。

- 2 つの素数 p, q $(2^{k-1} \leq p, q \leq 2^k - 1)$ を生成し、 $n := p^2 q$ を計算する。
- 整数 $e > 4$ (e.g., $e = 2^l, l = O(\log n)$) を選択する。

- $pLen := k$ を決定する。
- （ハッシュ）関数 H: $\{0,1\}^* \longrightarrow \{0,1\}^{pLen-1}$ を定め、、その識別番号を HID とする。HID = 1 は、本仕様書 9 章で例示するハッシュ関数とする。

注: $H(x)$ 自体ではなく $0||H(x)$ が署名作成と検証に必要なので、$H(x)$ は、以下のようなハッシュ関数 H' : $\{0,1\}^* \longrightarrow \{0,1\}^{pLen}$ を用いることによって得ることができる。まず、$H'(x)$ を計算し、$H'(x)$ の最上位ビットのみを 0 に変更する。これを $0||H(x)$ とする。

7.3 署名生成: $\mathcal{S}$

$\mathcal{S}$ の入力と出力は以下の通りである。

[入力] メッセージ $m \in \{0,1\}^{mLen}$ および公開鍵 $(n,e,HID,pLen) \in \mathbf{Z}^3$ と秘密鍵 $(p,q) \in \mathbf{Z}^2$。

[出力] 署名 $s \in \mathbf{Z}$.

$\mathcal{S}$ の処理は以下の通りである。

1. ランダムかつ一様に $(\boldsymbol{Z}/pq\boldsymbol{Z})\backslash p\mathbf{Z} := \{r \in \boldsymbol{Z}/pq\boldsymbol{Z} | \gcd(r,p) = 1\}$ から r を選ぶ。
2. $z := (0||H(m)||0^{2 \cdot pLen}) \in \{0,1\}^{3 \cdot pLen}$ と $\alpha := (\mathbf{I}[z] - r^e) \bmod n$ を求める。
3. 以下のようにして (w_0, w_1) を求める。

$$w_0 := \lceil \frac{\alpha}{pq} \rceil, \tag{5}$$

$$w_1 := w_0 \cdot pq - \alpha. \tag{6}$$

もし $w_1 \geq 2^{2pLen-1}$ であれば、Step 1 にもどる。(つまり, もし $\mathbf{B}_{2 \cdot pLen}[w_1]$ の最上位ビットが 1 の場合、Step 1 にもどる。)

4. $t := \frac{w_0}{er^{e-1}} \bmod p$, および $s := (r + tpq) \bmod n$ を計算する。
5. m の署名として s を出力する。

7.4 署名検証: $\mathcal{V}$

$\mathcal{V}$ の入力と出力は以下の通りである。

[入力] 署名 $s \in \mathbf{Z}$、文書 $m \in \{0,1\}^{pLen-1}$ および公開鍵 $(n, e, H, pLen) \in \mathbf{Z}^3$。

[出力] 検証結果 (「正当」もしくは「不正」)。

$\mathcal{V}$ の処理 は、以下の通りである。

- $$b := \mathbf{B}_{3 \cdot pLen}[s^e \bmod n]$$

  としたとき、以下の式が満足されるかどうかをチェックする：

  $$[b]^{pLen} = 0 || H(m). \tag{7}$$

- もし満足されれば, 「正当」(もしくは 1) を出力する。さもなければ、「不正」(もしくは 0) を出力する。

8 推奨パラメータ

以下では、 ESIGN 認証 のパラメータの推奨値を示す。

- k: 320 以上 (n のサイズを 960 ビット以上)
- e: 8 以上

以下、「ESIGN 認証自己評価書」で示した「性能評価」での代表的なパラメータ値を示す。

n のサイズを 1152 ビット、 $e = 2^5 (= 32)$ とする。(なお、実装評価では、$e = 1024$ としている。)

9 ハッシュ関数

既に述べたように、H が理想的なランダム関数であるとき ESIGN の安全性を証明することができる。一方、実際にこの署名アルゴリズムを実現する際には理想的なランダム関数の代わりに実用的な一方向性関数 (例えば SHA など) を用いる。ここでは、SHA を

用いて任意のサイズ（$hLen$ ビット）を出力する関数 H の一構成例を示す。この方法は、Bellare と Rogaway により示された方法である [2]。

$\mathrm{SHA}_\sigma(x)$ は x に SHA を適用して得られた 160 ビットの出力値を意味する。但し、160 ビットの "初期値" を $ABCDE = \sigma$ とする。$\mathrm{SHA}^l_\sigma(x)$ を $\mathrm{SHA}_\sigma(x)$ の先頭 l ビットとする。また、i を 32 ビットに符号化した値を $< i >$ とする。関数 H を以下のように定める。

$$H(x) := \mathrm{SHA}^{80}_\sigma(< 0 > ||x)||\mathrm{SHA}^{80}_\sigma(< 1 > ||x)|| \cdots ||\mathrm{SHA}^{L_l}_\sigma(< l > ||x),$$

ここで $l = \lfloor \frac{3k}{80} \rfloor$, かつ $L_l = hLen - 80l$.

参考文献

[1] Bellare, M. and Rogaway, P.: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols, Proc. of the First ACM Conference on Computer and Communications Security, pp.62–73 (1993).

[2] Bellare, M. and Rogaway, P. : Optimal Asymmetric Encryption, Proc. of Eurocrypt'94, LNCS 950, Springer-Verlag pp.92-111 (1995).

[3] Bellare, M. and Rogaway, P.: The Exact Security of Digital Signatures – How to Sign with RSA and Rabin, Proc. of Eurocrypt'96, LNCS 1070, Springer-Verlag, pp.399-416 (1996).

[4] Canetti, R., Goldreich, O. and Halevi, S.: The Random Oracle Methodology, Revisited, Proc. of STOC, ACM Press, pp.209–218 (1998).

[5] Damgård, I., Landrock, P., and Pomerance, C., "Average Case Error Estimates for the Strong Probable Prime Test", Matematics of Computation 61(1993), pp.177–194.

[6] Fujisaki, E. and Okamoto, T.: Security of Efficient Digital Signature Scheme TSH-ESIGN, manuscript (1998 November).

[7] S. Goldwasser, S. Micali and R. Rivest, "A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks," SIAM J. on Computing, 17, pp.281–308, 1988.

[8] IEEE P1363 Draft (D9), http://grouper.ieee.org/groups/1363/P1363/draft.html (1999).

[9] Menezes, A., van Oorschot, P., and Vanstone, S., "Handbook of Applied Cryptography", CRC Press, Boca Raton, Florida 1996.

[10] Okamoto, T.: A Fast Signature Scheme Based on Congruential Polynomial Operations, IEEE Trans. on Inform. Theory, IT-36, 1, pp.47-53 (1990).

[11] FIPS 180-1 "Secure Hash Standard", Federal Information Processing Standards Publication 180-1, U.S. Department of Commerce/N.I.S.T., National Technical Information Service, Springfield, Virginia, April 17 1995.