ESIGN-D Specification

NTT Information Sharing Platform Laboratories,
NTT Corporation

August 19, 2002
Contents

1 Introduction 3

2 Notation 4

3 Data types and conversions 4
   3.1 Integer-to.BitString Conversion(I2BSP) 4
   3.2 BitString-to-Integer Conversion(BS2IP) 5
   3.3 BitString-to-OctetString Conversion(BS2OSP) 6
   3.4 OctetString-to-BitString Conversion(OS2BSP) 6
   3.5 Integer-to-OctetString Conversion(I2OSP) 7
   3.6 OctetString-to-Integer Conversion(OS2IP) 7

4 Key types 8
   4.1 ESIGN-D private key 8
   4.2 ESIGN-D public key 8

5 Cryptographic primitives 8
   5.1 KGP-ESIGN-D 8
   5.2 SP-ESIGN-D 9
   5.3 VEP-ESIGN-D 10

6 Signature schemes with appendix 10
   6.1 SSA-ESIGN-D 10
      6.1.1 Signature generation operation 10
      6.1.2 Signature verification operation 11

7 Encoding methods 11
   7.1 EMSA-ESIGN-D 11
      7.1.1 Encoding operation 11
      7.1.2 Verification operation 12

8 Auxiliary techniques 12
   8.1 Hash functions 12
      8.1.1 SHA-1 12
   8.2 Mask generation functions 12
      8.2.1 MGF1 13

A Security requirements of parameters 15

B Recommended values of parameters 15
1 Introduction

This document provides a specification for implementing ESIGN-D. It is an improved version of
ESIGN-TSH specified in IEEE P1363a / D10 [1], which was suggested by [2].

ESIGN-D is secure against chosen message attacks (CMA) and ESIGN-TSH is secure against
single occurrence chosen message attacks (SO-CMA) that is weaker than CMA.

In this document, SSA-ESIGN-D-Sign and SSA-ESIGN-D-Verify describe the algo-
rithms of ESIGN-D. They consist of

• cryptographic primitives
  – Key generation primitive: KGP-ESIGN-D,
  – Signature primitive: SP-ESIGN-D,
  – Verification primitive: VP-ESIGN-D, and

• an encoding method
  – EMSA-ESIGN-D.
2 Notation

bit one of the two symbols 0 or 1.
bit string an ordered sequence of bits.
octet one of the integers from 0 to 255.
octet string an ordered sequence of octets.
\(R\) the set of real numbers.
\(Z\) the set of integers.
\(N\) the set of positive integers.
\(a := b\) assign \(b\) to \(a\).
\((B_0, B_1, \ldots, B_{i-1})_2\) a bit string of length \(i\), for example, \((0, 1, 0, 0)_2\).
\((M_0, M_1, \ldots, M_{i-1})_{256}\) an octet string of length \(i\), for example, \((170, 255, 0)_{256}\).
\(\{0, 1\}^i\) if \(i \in N\), then the set of all bit strings of length \(i\). if \(i = 0\), then empty bit string.
\(\{0, 1\}^*\) \(\bigcup_{i=0}^{\infty}\{0, 1\}^i\).
\(\{0, 1, \ldots, 255\}^i\) if \(i \in N\), then the set of all octet strings of length \(i\). if \(i = 0\), then empty octet string.
\(\{0, 1, \ldots, 255\}^*\) \(\bigcup_{i=0}^{\infty}\{0, 1, \ldots, 255\}^i\).
\(\|\) a concatenation operator for two bit strings or for two octet strings, for example, \((0, 1, 0, 0)_2 \| (1, 1, 0)_2 = (0, 1, 0, 0, 1, 1, 0)_2\) for bit strings, \((170, 255)_{256} \| (0, 20)_{256} = (170, 255, 0, 20)_{256}\) for octet strings. The operator is often omitted.
\([y]\) for \(y \in R\), the least integer greater than or equal to \(y\).
\([y]\) for \(y \in R\), the greatest integer less than or equal to \(y\).
\([X]_y\) for \(y \in N\), the rightmost \(y\) bits of a bit string \(X\).
GCD\((a, b)\) for \(a \in N\), \(b \in N\), the greatest common divisor of \(a\) and \(b\).
\(a \mid b\) for \(a \in Z\), \(b \in Z\), \(a\) divides \(b\).
\(a \mod m\) for \(a \in Z\), \(m \in N\), the least nonnegative integer \(b\) which satisfies \(m | (b - a)\).
\(a^{-1} \mod m\) for \(a \in Z\), \(m \in N\), the least nonnegative integer \(b\) which satisfies \(ab \mod m = 1\).

3 Data types and conversions

The schemes specified in this document involve operations using several different data types. Figure 1 illustrates which conversions are needed and where they are described.

3.1 Integer-to-BitString Conversion(I2BSP)

Integers should be converted to bit strings as described in this section. Informally, the idea is to represent the integer in binary. Formally, the conversion routine, I2BSP\((x, l)\), is specified as follows:

Input: \(x\) an integer to be converted, \(a\) nonnegative integer
\(l\) the bit length of the output, \(a\) nonnegative integer
Output: \(B\) a bit string of length \(l\) bits
Errors: INVALID
Steps:
\[ n = \left\lfloor \frac{l}{8} \right\rfloor, \quad M_i \in \{0, 1, \ldots, 255\} \]

Octet String
\[ M = M_0 M_1 \cdots M_{n-1} \]

Nonnegative Integer \( x \)

Bit String
\[ B = B_0 B_1 \cdots B_{l-1} \]
\[ B_i \in \{0, 1\} \]

Figure 1: Conversion between data types

1. If \( l = 0 \), output an empty bit string and stop.
2. If \( x \geq 2^l \), assert INVALID and stop.
3. Determine the \( x \)'s base-2 representation, \( x_i \in \{0, 1\} \) such that
   \[ x = x_{l-1}2^{l-1} + x_{l-2}2^{l-2} + \cdots + x_1 2 + x_0. \]
4. For \( 0 \leq i \leq l-1 \), set \( B_i := x_{l-i} \), and let
   \[ B := B_0 B_1 \cdots B_{l-1}. \]
5. Output \( B \).

3.2 BitString-to-Integer Conversion (BS2IP)

Bit strings should be converted to integers as described in this section. Informally, the idea is simply to view the bit string as the base-2 representation of the integer. Formally, the conversion routine, \( BS2IP(B, l) \), is specified as follows:

**Input:**
- \( B \) a bit string to be converted
- \( l \) the bit length of \( B \), a nonnegative integer

**Output:**
- \( x \) a nonnegative integer

**Steps:**
Convert $B = B_0 B_1 \cdots B_{l-1}$ to an integer $x$ as follows:

1. If $l = 0$, output 0 and stop.
2. View each $B_i$ as an integer in $\{0, 1\}$, set $x_i := B_i$ for $0 \leq i \leq l - 1$, and let
   
   $$x := \sum_{i=0}^{l-1} 2^{l-i-1} x_i.$$ 

3. Output $x$.

### 3.3 BitString-to-OctetString Conversion (BS2OSP)

Bit strings should be converted to octet strings as described in this section. Informally, the idea is to pad the bit string with 0’s on the left to make its length a multiple of 8, then chop the result up into octets. Formally, the conversion routine, BS2OSP($B, l$), is specified as follows:

**Input:** $B$ a bit string to be converted
   
   $l$ the bit length of $B$, a nonnegative integer

**Output:** $M$ an octet string of length $n = \left\lceil \frac{l}{8} \right\rceil$ octets

**Steps:**

Convert the bit string $B = B_0 B_1 \cdots B_{l-1}$ to an octet string $M = M_0 M_1 \cdots M_{n-1}$ as follows:

1. If $l = 0$, output an empty octet string and stop.
2. For $0 < i \leq n - 1$, let:
   
   $$M_i := B_{i - 8 - 8(n-1-i)} B_{i - 7 - 8(n-1-i)} \cdots B_{i - 1 - 8(n-1-i)}.$$ 

3. Set
   
   $$M_0 := \begin{cases} \neg B_0 \neg B_1 \cdots \neg B_7 & (8n - l = 0) \\ \neg Z \neg B_0 \cdots \neg B_{7 + 8n} & (8n - l \neq 0) \end{cases},$$

   where $Z$ is an all zero bit string of length $8n - l$ ($Z = (0,0, \ldots, 0)_2$).

4. Output $M$.

### 3.4 OctetString-to-BitString Conversion (OS2BSP)

Octet strings should be converted to bit strings as described in this section. Informally, the idea is simply to view the octet string as a bit string. Formally, the conversion routine, OS2BSP($M, l$), is specified as follows:

**Input:** $M$ an octet string of length $n = \left\lceil \frac{l}{8} \right\rceil$ octets to be converted
   
   $l$ the bit length of the output, a nonnegative integer

**Output:** $B$ a bit string of length $l$ bits

**Steps:**

Convert the octet string $M = M_0 M_1 \cdots M_{n-1}$ to a bit string $B = B_0 B_1 \cdots B_{l-1}$ as follows:
1. If \( l = 0 \), output an empty bit string and stop.

2. For \( 0 < i \leq n - 1 \), set:
   \[
   B_{i} := s_{(n-1-i)} B_{i-1} B_{i-2} B_{i-3} \cdots B_{i-(n-1-i)} := M_i.
   \]

3. For \( 0 \leq j \leq l + 7 - 8n \), set \( B_j := Z_j + s_{n-1-i} \), where \( Z_0 Z_1 \cdots Z_7 := M_0 \).

4. Output \( B \).

3.5 Integer-to-OctetString Conversion (I2OSP)

Integers should be converted to octet strings as described in this section. Informally, the idea is to represent the integer in binary and then convert the resulting bit string to an octet string. Formally, the conversion routine, \( \text{I2OSP}(x, l) \), is specified as follows:

**Input:** \( x \) an integer to be converted, \( l \) a nonnegative integer

**Output:** \( M \) an octet string of length \( n = \left\lceil \frac{l}{8} \right\rceil \) octets

**Errors:** INVALID

**Steps:**

1. If \( l = 0 \), output an empty octet string and stop.

2. If \( x \geq 2^l \), assert INVALID and stop.

3. Determine the \( x \)'s base-256 representation, \( x_i \in \{0, \ldots, 255\} \) such that
   \[
   x = x_{n-1} 2^{8(n-1)} + x_{n-2} 2^{8(n-2)} + \cdots + x_1 2^8 + x_0.
   \]

4. For \( 0 \leq i \leq n - 1 \), set \( M_i := x_{n-1-i} \), and let
   \[
   M := M_0 M_1 \cdots M_{n-1}.
   \]

5. Output \( M \).

3.6 OctetString-to-Integer Conversion (OS2IP)

Octet strings should be converted to integers as described in this section. Informally, the idea is simply to view the octet string as the base-256 representation of the integer. Formally, the conversion routine, \( \text{OS2IP}(M, l) \), is specified as follows:

**Input:** \( M \) an octet string of length \( n = \left\lceil \frac{l}{8} \right\rceil \) octets to be converted

\( l \) the bit length of \( x \), a nonnegative integer

**Output:** \( x \) a nonnegative integer

**Steps:**

1. Convert \( M = M_0 M_1 \cdots M_{n-1} \) to an integer, \( x \), as follows:

2. View each \( M_i \) as an integer in \( \{0, \ldots, 255\} \), set \( x_i := M_i \) for \( 0 \leq i \leq l - 1 \), and let
   \[
   x := \sum_{i=0}^{n-1} 2^{8(n-1-i)} x_i \mod 2^l.
   \]

3. Output \( x \).
4 Key types

In this section, two types of keys are defined: ESIGN-D private key and ESIGN-D public key, both of which are employed in the primitives and schemes.

4.1 ESIGN-D private key

An ESIGN-D private key is the 6-tuple \((p, q, n, p\text{Len}, \epsilon, \text{SEED})\), where the components have the following meanings:

- \(p\), the first factor, a prime number, \(2^\text{p\text{Len}-1} < p < 2^\text{p\text{Len}}\).
- \(q\), the second factor, a prime number, \(2^\text{p\text{Len}-1} < q < 2^\text{p\text{Len}}\) and \(q \neq p\).
- \(n\), the modulus, a positive integer, \(2^{3\text{p\text{Len}-1}} < n < 2^{3\text{p\text{Len}}}\) and \(n = p^2 q\).
- \(p\text{Len}\), the security parameter, a positive integer.
- \(\epsilon\), the exponent, a positive integer.
- \(\text{SEED}\), the index to a pseudo-random function, a nonnegative integer, \(0 \leq \text{SEED} < 2^{2\text{p\text{Len}}}\).

Valid ESIGN-D private key We call a private key \((p, q, n, p\text{Len}, \epsilon, \text{SEED})\) “valid” if it satisfies all the above conditions.

4.2 ESIGN-D public key

An ESIGN-D public key is the 3-tuple \((n, p\text{Len}, \epsilon)\), where the components have the following meanings:

- \(n\), the modulus, a positive integer, \(2^{3\text{p\text{Len}-1}} < n < 2^{3\text{p\text{Len}}}\) and \(n = p^2 q\).
- \(p\text{Len}\), the security parameter, a positive integer.
- \(\epsilon\), the exponent, a positive integer.

Valid ESIGN-D public key We call a public key \((n, p\text{Len}, \epsilon)\) “valid” if it satisfies all the above conditions.

5 Cryptographic primitives

In this section, three cryptographic primitives are specified.

5.1 KGP-ESIGN-D

KGP-ESIGN-D\((k, \epsilon)\) is defined as follows:

Input: \(k\) security parameter, a positive integer
\(\epsilon\) public exponent, a positive integer

Output: \(PK\) ESIGN-D public key \((n, p\text{Len}, \epsilon)\)
\(SK\) ESIGN-D private key \((p, q, n, p\text{Len}, \epsilon, \text{SEED})\)

Steps:
1. Choose two odd primes $p, q$ such that $2^{k-1} < p < 2^k$, $2^{k-1} < q < 2^k$, $p \neq q$, $2^{3k-1} \leq p^2 q < 2^{3k}$.

2. Compute $n := p^2 q$.


4. Generate a random integer $\text{seed} \in \{0, 1, \ldots, 2^{2k} - 1\}$

5. Output $PK = (n, pLen, e)$, $SK = (p, q, n, pLen, e, \text{seed})$.

5.2 SP-ESIGN-D

SP-ESIGN-D$(SK, f)$ is defined as follows:

**System parameters:**
- $Hash$ hash function
- $hLen$ length in bits of the hash function output, a positive integer
- $MGF$ mask generation function

**Input:**
- $SK$ ESIGN-D private key $(p, q, n, pLen, e, \text{seed})$
- $f$ message representative, an integer, $0 \leq f < 2^{pLen-1}$

**Output:**
- $s$ signature representative, an integer, $0 \leq s < n$

**Errors:** INVALID

**Steps:**

1. If the message representative $f$ does not satisfy $0 \leq f < 2^{pLen-1}$, assert INVALID and stop.

2. Compute $z := f \cdot 2^{pLen}$.

3. Set $c := 0$.


5. If $c \geq 2^{32}$, assert INVALID and stop.

6. Compute

$$r := \text{OS2IP}(\text{MGF}(\text{I2OSP}($\text{seed}$, 2pLen)) || \text{I2OSP}(f, pLen - 1) || \text{I2OSP}(c, 32), 2pLen), 2pLen).$$

7. If $r \geq pq$, go back to step 4.

8. Compute $\alpha := (z - r^e) \mod n$.

9. Let $(w_0, w_1)$ be

$$w_0 := \left\lfloor \frac{\alpha}{pq} \right\rfloor,$$

$$w_1 := w_0pq - \alpha.$$

10. If $w_1 \geq 2^{pLen-1}$, go back to step 4.

11. If $r \mod p = 0$, go back to step 4.

12. Let $t := w_0(\alpha^{r-1})^{-1} \mod p$, and compute $s := r + tpq$.

13. Output $s$. 

9
5.3 VP-ESIGN-D

VP-ESIGN-D($PK$, $s$) is defined as follows:

**Input:** $PK$  
ESIGN-D public key $(n, pLen, e)$  
$s$  
signature representative, an integer, $0 \leq s < n$

**Output:** $f$  
message representative, an integer, $0 \leq f < 2^{pLen-1}$

**Errors:** INVALID

**Steps:**

1. If the signature representative $s$ does not satisfy $0 \leq s < n$, assert INVALID and stop.
2. Compute $t := s^e \mod n$.
3. Compute $f := \lfloor \frac{t}{2^{pLen}} \rfloor$.
4. If $f$ does not satisfy $0 \leq f < 2^{pLen-1}$, assert INVALID and stop.
5. Output $f$.

**Note:** If $n \leq 2^{3pLen-1}$ or $n \geq 2^{3pLen}$, the performance of this function is not guaranteed.

6 Signature schemes with appendix

6.1 SSA-ESIGN-D

SSA-ESIGN-D combines the SP-ESIGN-D and VP-ESIGN-D primitives with the EMSA-ESIGN-D encoding method.

6.1.1 Signature generation operation

SSA-ESIGN-D-Sign($SK$, $M$) is defined as follows:

**Input:** $SK$  
signer's ESIGN-D private key  
$M$  
message to be signed, an octet string

**Output:** $s$  
signature representative, an integer, $0 \leq s < n$

**Errors:** INVALID

**Steps:**

1. Apply the EMSA-ESIGN-D-Encode operation (Section 7.1.1) to the message $M$ to produce an integer message representative $f$:

   $$f := \text{EMSA-ESIGN-D-Encode}(M, pLen - 1).$$

   If the encoding operation asserts INVALID, then assert INVALID and stop.
2. Apply the SP-ESIGN-D signature primitive (Section 5.2) to the private key $SK$ and the message representative $f$ to produce an integer signature representative $s$:

   $$s := \text{SP-ESIGN-D}(SK, f).$$

3. Output the signature representative $s$.  

10
6.1.2 Signature verification operation

SSA-ESIGN-D-VERIFY$(PK, M, s)$ is defined as follows:

**Input:**
- $PK$  signer’s ESIGN-D public key
- $M$  message whose signature is to be verified, an octet string
- $s$  signature to be verified, an integer, $0 \leq s < n$

**Output:** VALID_SIGNATURE or INVALID_SIGNATURE

**Steps:**

1. Apply the VP-ESIGN-D verification primitive (Section 5.3) to the public key $PK$ and the signature representative $s$ to produce an integer message representative $f'$:

   $$f' := \text{VP-ESIGN-D}(PK, s).$$

   If the VP-ESIGN-D primitive asserts INVALID, then output INVALID_SIGNATURE and stop.

2. Apply the EMSA-ESIGN-D-VERIFY operation (Section 7.1.2) to the message $M$ and the message representative $f'$ to determine whether they are consistent:

   $$\text{Result} := \text{EMSA-ESIGN-D-VERIFY}(M, f', pLen - 1).$$

   If Result is CONSISTENT, then output VALID_SIGNATURE.
   Otherwise, output INVALID_SIGNATURE.

7 Encoding methods

7.1 EMSA-ESIGN-D

7.1.1 Encoding operation

EMSA-ESIGN-D-ENCODE$(M, l)$ is defined as follows:

**System parameters:**
- $Hash$  hash function
- $hLen$  length in bits of the hash function output, a positive integer
- $MGF$  mask generation function

**Input:**
- $M$  message to be encoded, an octet string
- $l$  the bit length of the encoded message, a positive integer

**Output:**
- $f$  encoded message representative, a nonnegative integer

**Errors:** INVALID

**Steps:**

1. If the length of $M$ is greater than the input limitation for the hash function, assert INVALID and stop.

2. Apply the hash function to the message $M$ to produce a hash value $H$ of length $\left\lceil \frac{hLen}{8} \right\rceil$ octets:

   $$H := \text{Hash}(M).$$
3. Apply the mask generation function to $H$ to produce an octet string $T$ of length $\lceil \frac{l}{m} \rceil$ octets:

$$T := MGF(H, l).$$

4. Convert the octet string $T$ to an integer $f$:

$$f := OS2IP(T, l).$$

5. Output $f$.

### 7.1.2 Verification operation

$EMSA-\text{ESIGN-D-VVerify}(M, f, l)$ is defined as follows:

<table>
<thead>
<tr>
<th>System parameters:</th>
<th>Hash</th>
<th>hash function</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$hLen$</td>
<td>length in bits of the hash function output, a positive integer</td>
</tr>
<tr>
<td></td>
<td>$MGF$</td>
<td>mask generation function</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input:</th>
<th>$M$</th>
<th>message to be encoded, an octet string</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$f$</td>
<td>encoded message representative, a nonnegative integer</td>
</tr>
<tr>
<td></td>
<td>$l$</td>
<td>the bit length of $f$, a positive integer</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output:</th>
<th>CONSISTENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Errors:</td>
<td>INVALID</td>
</tr>
<tr>
<td>Steps:</td>
<td></td>
</tr>
</tbody>
</table>

1. Produce an integer message representative $f'$:

$$f' := EMSA-\text{ESIGN-D-Encode}(M, l).$$

2. If $f$ is equal to $f'$, then output CONSISTENT. Otherwise assert INVALID.

## 8 Auxiliary techniques

This section gives several examples of the techniques that support the functions described in this document.

### 8.1 Hash functions

One hash function is recommended for the encoding methods in this document: SHA-1.

#### 8.1.1 SHA-1

SHA-1 is defined in FIPS PUB 180-1 [3]. The output length of SHA-1 is 160 bits, and the operation block size is 512 bits.

### 8.2 Mask generation functions

One mask generation function is recommended for the encoding methods in this document: MGF1 [1].
8.2.1 MGF1

MGF1 is a mask generation function based on a hash function.
MGF1(M,l) is defined as follows:

System parameters:  
<table>
<thead>
<tr>
<th>Hash function</th>
</tr>
</thead>
<tbody>
<tr>
<td>hashLen</td>
</tr>
</tbody>
</table>

Input:  
| M   | seed from which mask is generated, | an octet string |
| l   | the bit length of the output, | a positive integer |

Output:  
| mask | mask, an octet string of length \( \left\lceil \frac{l}{8} \right\rceil \) octets |

Errors:  
| INVALID |

Steps:

1. Let \( l_0 \) be the bit length of \( M \). If \( l_0 + 32 \) is greater than the input limitation for the hash function, assert INVALID and stop.

2. Let \( cThreshold := \left\lceil \frac{l}{\text{hashLen}} \right\rceil \).

3. Let \( M' \) be the empty octet string.

4. Let \( counter := 0 \).

   (a) Convert the integer \( counter \) to an octet string \( C \) of length 32 bits:

   \[
   C := I2OSP(counter, 32).
   \]

   (b) Concatenate \( M \) and \( C \), and apply the hash function to the result to produce a hash value \( H \) of length \( \left\lceil \frac{\text{hashLen}}{8} \right\rceil \) octets:

   \[
   H := \text{Hash}(M \| C).
   \]

   (c) Concatenate \( M' \) and \( H \) to the octet string \( M' \):

   \[
   M' := M' \| H.
   \]

   (d) Let \( counter := counter + 1 \). If \( counter < cThreshold \), go back to step 4a.

5. Let \( mask \) be the leftmost \( \left\lceil \frac{l}{8} \right\rceil \) octets of the octet string \( M' \):

   \[
   mask := M'_0M'_1 \cdots M'_{\left\lceil \frac{l}{8} \right\rceil - 1}.
   \]

6. Output \( mask \).

References


Appendix

A Security requirements of parameters

Security requirements of ESIGN-D parameters are the following:

\[ k \geq 342 \quad \text{(bit length of } n \geq 1024) \]
\[ 2^{k/4} \geq \epsilon \geq \frac{3k}{2} \]

B Recommended values of parameters

Recommended values of ESIGN-D parameters are the following:

\[ k = 512 \text{ or } 1024 \quad \text{(bit length of } n = 1536 \text{ or } 3072) \]
\[ \epsilon = 65537 \]
\[ Hash = SHA-1 \]
\[ hLen = 160 \]
\[ MGF = MGF1 \quad \text{(SHA-1, } hashLen = 160) \]