## Contents

1 Introduction

1.1 Organization

1.2 Outline of the JCATT

2 Scope

2.1 Public-key Cryptography

2.1.1 Digital Signature

2.1.2 Confidentiality

3 Tests Description of Public-Key Cryptography

3.1 Digital Signature

3.1.1 DSA (FIPS 186-2)

3.1.1.1 The Domain Parameter Generation Test

3.1.1.2 The Domain Parameter Validation Test

3.1.1.3 Key Pair Generation Test

3.1.1.4 Signature Generation Test

3.1.1.5 Signature Verification Test

3.1.2 ECDSA

3.1.2.1 The Domain Parameter Generation Test

3.1.2.2 The Domain Parameter Validation Test

3.1.2.3 Key Pair Generation Test

3.1.2.4 Public Key Validation Test

3.1.2.5 Signature Generation Test

3.1.2.6 Signature Verification Test

3.1.3 RSASSA-PKCS1-v1_5

3.1.3.1 Key Pair Generation Test

3.1.3.2 Signature Generation Test

3.1.3.3 Signature Verification Test

3.1.4 RSASSA-PSS

3.1.4.1 Key Pair Generation Test

3.1.4.2 Signature Generation Test

3.1.4.3 Signature Verification Test

3.2 Confidentiality

3.2.1 RSA-OAEP

3.2.1.1 Key Pair Generation Test

3.2.1.2 The Encryption Test

3.2.1.3 The Decryption Test

3.2.2 RSAES-PKCS1-v1_5

3.2.2.1 Key Pair Generation Test

3.2.2.2 The Encryption Test

3.2.2.3 The Decryption Test

4 Conditions for Issuing Cryptographic Algorithm Validation Certificate

4.1 Details of Conditions

References
1 Introduction

This document describes the specifications of cryptographic algorithm implementation testing of the public-key cryptographic algorithms.

1.1 Organization

Section 2 specifies the cryptographic algorithms that are in the scope of this document. Section 3 provides an overview of the tests of each algorithm that make up the JCATT. Section 4 provides the conditions for issuing cryptographic algorithm validation certificate.

The following acronyms are used throughout this document.

- JCATT: Japan Cryptographic Algorithm implementation Testing Tool
- IUT: Implementation Under Test

1.2 Outline of the JCATT

The Japan Cryptographic Algorithm implementation Testing Tool is designed:

- to test conformance to the cryptographic algorithm specifications,
- to test each function of the cryptographic algorithm — for example, signature generation, verification, key generation, etc for digital signature algorithm. —
- to allow the testing of an IUT at locations remote to the JCATT. The JCATT and the IUT communicate data via REQUEST and RESPONSE files.

Once configuration information has been provided, appropriate REQUEST files will be generated. REQUEST files are the means by which test data is communicated to the IUT. The IUT is used to process the data in the REQUEST file, and the resulting data is placed in a RESPONSE file. The data in the RESPONSE file is then verified.

The specification of the file format is available in ref. [2] and the sample files are available in ref. [3].

Figure 1.1: The Workflow of the Cryptographic Algorithm Implementation Testing
2 Scope

This document specifies the tests required to validate IUTs implementing the following cryptographic algorithms.

2.1 Public-key Cryptography

2.1.1 Digital Signature
• DSA (FIPS 186-2)
• ECDSA
• RSASSA-PKCS1-v1_5
• RSASSA-PSS

2.1.2 Confidentiality
• RSA-OAEP
• RSAES-PKCS1-v1_5
3 Tests Description of Public-Key Cryptography

3.1 Digital Signature

This section provides the tests required to validate the following algorithms of digital signature:

- DSA (FIPS 186-2).
- ECDSA
- RSASSA-PKCS1-v1_5
- RSASSA-PSS

3.1.1 DSA (FIPS 186-2)

The following functions of DSA (FIPS 186-2) are tested:

- Domain Parameter Generation
- Domain Parameter Validation
- Key-pair Generation
- Signature Generation
- Signature Verification

3.1.1.1 The Domain Parameter Generation Test

The following tests are performed for the domain parameter generation:

- Input the SEED and counter produced by IUT to JCATT. JCATT calculates two prime numbers \( p' \) and \( q' \) according to the algorithms described in FIPS 186-2 Appendix 2. \( p \) and \( q \) generated by IUT shall be equal to \( p' \) and \( q' \) respectively.
- \( g^q \equiv 1 \mod p \)
- IUT generates a number of domain parameters \((p,q,g)\), where the number is defined in section 4. All of these domain parameters shall be mutually different.

In addition, the validity of \( p, q, \) counter, SEED described in FIPS 186-2 Appendix 2 is tested.

3.1.1.2 The Domain Parameter Validation Test

The following domain parameter validation tests are performed:

- Domain parameters that have passed the test described in the previous subsection are given to IUT. IUT shall then return “Valid” as the validation result.
- Domain parameters that have failed the test described in the previous subsection are given to IUT. IUT shall then return “Invalid” as the validation result.

3.1.1.3 Key Pair Generation Test

Key pair generation test is performed by inspecting the following conditions are met:

- \( y \equiv g^x \mod p \)
- \( 1 \leq x \leq q - 1, 2 \leq y \leq p - 2 \)
- \( y^q \equiv 1 \mod p \)
- All of multiple key pairs generated by IUT shall be mutually different, where the number of key pairs is defined in section 4.
3.1.1.4 Signature Generation Test

The following signature generation tests are performed:

- JCATT provides IUT the private key $x$ and the messages to be signed. IUT generates the signatures for the messages provided by JCATT. JCATT validates the signatures. The signatures generated by IUT shall pass the verification test by JCATT.
- IUT generates multiple signatures, where the number of signatures is defined in section 4. for the same plaintext and the same private key. All of the signatures shall be mutually different.

3.1.1.5 Signature Verification Test

The following signature verification tests are performed:

- The public key $y$, the messages, the hash function, and the appropriate signatures for those parameters are given to IUT. IUT shall then return “Pass” as the verification result.
- The public key $y$, the messages, and the signatures for those parameters, one of which is altered, are given to IUT. IUT shall then return “Fail” as the verification result.
3.1.2 ECDSA

The following functions of ECDSA are tested:

- Domain Parameter Generation
- Domain Parameter Validation
- Key Pair Generation
- Public Key Validation Test
- Signature Generation
- Signature Verification

3.1.2.1 The Domain Parameter Generation Test

The following tests are performed for the domain parameter generation:

**For the Domain Parameters over \(\mathbb{F}_p\)**

The test case 1 or the test case 2 is performed depending on whether the elliptic curve is chosen verifiably at random. The test case 1 is performed by default.

**The test case 1 (The default test case)**

**Elliptic curves over \(\mathbb{F}_p\)**

- The bit length of \(n\) is 160 or more.
- \(4a^3 + 27b^2 \not\equiv 0 \mod p\)
- \(a, b, x_G, y_G\) are integers equal or greater than 0, equal or smaller than \(p - 1\).
- \(y_G^2 \equiv x_G^3 + ax_G + b \mod p\)
- \(n\) is prime.
- \(p\) is prime.
- \(h \leq 4\) and \(h = \left\lfloor (\sqrt{p} + 1)^2 / n \right\rfloor\).
- \(nG = \mathcal{O}\).
- \(q^B \not\equiv 1 \mod n\) for any \(B\) such that \(1 \leq B < 20\).
- \(nh \neq p\).
- All of the domain parameters generated by IUT are mutually different.

**The test case 2 (The test case for curves chosen verifiably at random)**

**Elliptic curves over \(\mathbb{F}_p\)**

- For IUT that outputs the SEED, JCATT validates the curve is chosen “verifiably at random” as described in ANSI X9.62 A.3.3.
- The bit length of \(n\) is 160 or more.
- \(4a^3 + 27b^2 \not\equiv 0 \mod p\)
- \(a, b, x_G, y_G\) are integers equal or greater than 0, equal or smaller than \(p - 1\).
- \(y_G^2 \equiv x_G^3 + ax_G + b \mod p\)
- \(n\) is prime.
- \(p\) is prime.
- \(h \leq 4\) and \(h = \left\lfloor (\sqrt{p} + 1)^2 / n \right\rfloor\).
- \(nG = \mathcal{O}\).
- \(q^B \not\equiv 1 \mod n\) for any \(B\) such that \(1 \leq B < 20\).
- \(nh \neq p\).
- All of the domain parameters generated by IUT are mutually different. The number of domain parameters to be generated is defined in section 4.
For the Domain Parameters over $\mathbb{F}_{2^m}$

The test case 1 or the test case 2 is performed depending on whether the elliptic curve is chosen verifiably at random. The test case 1 is performed by default.

**The test case 1** (The default test case)

**Elliptic curves over $\mathbb{F}_{2^m}$**

- The bit length of $n$ is 160 or more.
- $f(x)$ is an irreducible binary polynomial of degree $m$.
- $a, b, x_G, y_G$ are binary polynomials of degree $m - 1$ or less.
- $b \neq 0$ in $\mathbb{F}_{2^m}$.
- $y_G^2 + x_G y_G = x_G^3 + a x_G^2 + b$ in $\mathbb{F}_{2^m}$.
- $n$ is prime.
- $h \leq 4$ and $h = \lfloor (\sqrt{2^m} + 1)^2 / n \rfloor$.
- $nG = O'$.
- $2^{mb} \neq 1 \mod n$ for any $B$ such that $1 \leq B < 20$.
- $nh \neq 2^m$.
- All of the domain parameters generated by IUT are mutually different. The number of domain parameters to be generated is defined in section 4.

**The test case 2** (The test case for curves chosen verifiably at random)

**Elliptic curves over $\mathbb{F}_{2^m}$**

- For IUT that outputs the SEED, JCAT validates the curve is chosen “verifiably at random” as described in ANSI X9.62 A.3.3.
- The bit length of $n$ is 160 or more.
- $f(x)$ is an irreducible binary polynomial of degree $m$.
- $a, b, x_G, y_G$ are binary polynomials of degree $m - 1$ or less.
- $b \neq 0$ in $\mathbb{F}_{2^m}$.
- $y_G^2 + x_G y_G = x_G^3 + a x_G^2 + b$ in $\mathbb{F}_{2^m}$.
- $n$ is prime.
- $h \leq 4$ and $h = \lfloor (\sqrt{2^m} + 1)^2 / n \rfloor$.
- $nG = O'$.
- $2^{mb} \neq 1 \mod n$ for any $B$ such that $1 \leq B < 20$.
- $nh \neq 2^m$.
- All of the domain parameters generated by IUT are mutually different. The number of domain parameters to be generated is defined in section 4.

### 3.1.2.2 The Domain Parameter Validation Test

There are two types of tests for the domain parameter validation test: test case 1 and test case 2. The test case 1 is performed by default.

**The test case 1** (Default test case)

- Domain parameters that have passed the test described in the previous section are given to IUT. IUT shall then return “Valid” as the validation result.
- Domain parameters that have failed the test described in the previous section are given to IUT. IUT shall then return “Invalid” as the validation result.
The test case 2 (The test case for curves chosen verifiably at random)

- If IUT is capable of validating whether an elliptic curve is chosen verifiably at random as described in ANSI X9.62 A.3.3, IUT shall successfully validate valid domain parameters given by JCA TT that contain the SEED, and the validation shall fail for invalid domain parameters given by JCA TT.

3.1.2.3 Key Pair Generation Test

The key pair generation test is performed by inspecting whether the following conditions are met:

For the Domain Parameters over $\mathbb{F}_p$

- $Q \neq \mathcal{O}$.
- $y_Q^2 \equiv x_Q^3 + ax_Q + b \mod p$.
- $nQ = \mathcal{O}$.
- $Q = dG$.
- All of the key pairs generated by IUT are mutually different. The number of key pairs to be generated is defined in section 4.

For the Domain Parameters over $\mathbb{F}_{2^m}$

- $Q \neq \mathcal{O}$.
- $y_Q^2 + x_Qy_Q = x_Q^3 + ax_Q^2 + b \in \mathbb{F}_{2^m}$.
- $nQ = \mathcal{O}$.
- $Q = dG$.
- All of the key pairs generated by IUT are mutually different. The number of key pairs to be generated is defined in section 4.

3.1.2.4 Public Key Validation Test

The following public key validation tests are performed:

- Public keys that have passed the test described in the previous section are given to IUT. IUT shall then return “Valid” as the validation result.
- Public keys that have failed the test described in the previous section are given to IUT. IUT shall then return “Invalid” as the validation result.

3.1.2.5 Signature Generation Test

The following signature generation tests are performed:

- The signature generated by IUT for the private key $d$ and the message given by JCA TT shall be successfully validated by JCA TT.
- All of the generated signatures for the same plaintext and the same private key shall be mutually different. The number of signatures to be generated is defined in section 4.

The hash function is selected from SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, or RIPEMD-160.
3.1.2.6 Signature Verification Test

The following signature verification test is performed:

- The public key $Q$, the messages, the hash function, and the appropriate signatures for those parameters are given to IUT. IUT shall then return “Pass” as the verification result.
- The public key $Q$, the messages, and the signatures for those parameters, one of which is altered, are given to IUT. IUT shall then return “Fail” as the verification result.

The hash function is selected from SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, or RIPEMD-160.
3.1.3 RSASSA-PKCS1-v1.5

There are following functions to be tested for RSASSA-PKCS1-v1.5.

- Key Pair Generation
- Signature Generation
- Signature Verification

3.1.3.1 Key Pair Generation Test

Private keys for RSA algorithms(RSAES-PKCS1-v1.5, RSA-OAEP, RSASSA-PKCS1-v1.5, and RSASSA-PSS) described in PKCS#1[1] are classified in two categories depending on whether Chinese Reminder Theorem(CRT) is used in decryption and signature generation where the private key is used for computation. A private key is a pair of \((n,d)\) when CRT is not used, and is a quintet of \((p,q,dP,dQ,qInv)\) when CRT is used. Therefore, JCATT offers two types of test for the key pair generation test depending on whether IUT’s key pair generation function produces \(dP,dQ,qInv\) for CRT.

**Key Pair Generation Test without CRT**

- The bit length of \(n\) is as specified.
- The bit lengths of \(p\) and \(q\) are equal.
- \(p\) is prime.
- \(q\) is prime.
- \(n = pq\).
- \(e \cdot d \equiv 1 \mod \lambda(n)\).
- All of the key pairs generated by IUT are mutually different. The number of key pairs is defined in section 4

**Key Pair Generation Test with CRT**

- The bit length of \(n\) is as specified.
- The bit lengths of \(p\) and \(q\) are equal.
- \(p\) is prime.
- \(q\) is prime.
- \(n = pq\).
- \(e \cdot d \equiv 1 \mod \lambda(n)\).
- \(e \cdot dP \equiv 1 \mod p - 1\).
- \(e \cdot dQ \equiv 1 \mod q - 1\).
- \(q \cdot qInv \equiv 1 \mod p\).
- All of the key pairs generated by IUT are mutually different. The number of key pairs is defined in section 4

In the description above, \(\lambda(n)\) is the least common multiple of \(p - 1\) and \(q - 1\).

3.1.3.2 Signature Generation Test

The following signature generation test is performed:

- IUT shall generate the valid signatures for the private keys \((n,d)\) or \((p,q,dP,dQ,qInv)\), the messages, and the hash function.

The hash function is selected from SHA-1, SHA-256, SHA-384, or SHA-512.
3.1.3.3 Signature Verification Test

The following signature verification tests are performed:

- The public key \((n,e)\), the messages, the hash function, and the appropriate signatures for those parameters are given to IUT. IUT shall then return “Pass” as the verification result.
- The public key \((n,e)\), the messages, and the signatures for those parameters, one of which is altered, are given to IUT. IUT shall then return “Fail” as the verification result.

The hash function is selected from SHA-1, SHA-256, SHA-384, or SHA-512.
3.1.4 RSASSA-PSS

There are the following functions to be tested for RSASSA-PSS.

- Key Pair Generation
- Signature Generation
- Signature Verification

3.1.4.1 Key Pair Generation Test

The test is performed in the same way as the key pair generation test described in described in section 3.1.3.

3.1.4.2 Signature Generation Test

For the signature generation test, the following Test case 1, Test case 2, and Test case 3 are available. The Test case 1 is mandatory for validation and is performed by default.

**Test case 1** (The default and mandatory test)

- The signatures for the private keys \((n, d)\) or \((p, q, dP, dQ, qInv)\) and the messages given by JCA TT generated by IUT shall be successfully validated by JCA TT.
- Under the condition that the length of salt is not zero, all of the signatures generated by IUT for the same message and same private key shall be mutually different. The number of signatures to be generated is defined in section 4.

**Test case 2** (Optional. Known Answer Test)

- IUT shall generate valid signatures for the private keys \((n, d)\) or \((p, q, dP, dQ, qInv)\), the messages, the hash function, pseudorandom number generator function, and the seed of PRNG given by JCA TT.

**Test case 3** (Optional. Known Answer Test with salt provided)

- IUT shall generate valid signatures for the private keys \((n, d)\) or \((p, q, dP, dQ, qInv)\), the messages, the hash function, and the salts.

The hash function is selected from SHA-1, SHA-256, SHA-384, or SHA-512. When the test case 2 is performed, the pseudorandom number generator function is selected from the following:

- PRNG based on SHA-1 in ANSI X9.42-2001 Annex C.1
- PRNG based on SHA-1 for general purpose in FIPS 186-2 (+ change notice 1) Appendix 3.1
- PRNG based on SHA-1 for general purpose in FIPS 186-2 (+ change notice 1) revised Appendix 3.1

3.1.4.3 Signature Verification Test

The test is performed in the same way as the signature verification test for RSASSA-PKCS-v1_5 described in section 3.1.3.
3.2 Confidentiality

This section provides the description of tests for public-key cryptographic algorithm (for confidentiality) RSA-OAEP and RSAES-PKCS1-v1_5.

3.2.1 RSA-OAEP

There are the following functions to be tested for RSA-OAEP.

- Key Pair Generation
- Encryption
- Decryption

3.2.1.1 Key Pair Generation Test

The test is performed in the same way as the key pair generation test described in section 3.1.3.

3.2.1.2 The Encryption Test

For the encryption test, the following Test case 1, Test case 2, and Test case 3 are available. The Test case 1 is mandatory for validation and is performed by default.

Test case 1 (The default and mandatory test)

- IUT encrypts the plaintext with the public key \((n, e)\), the hash function, the mask generation function \(MGF\) and the label \(L\) given by JCATT. JCATT then decrypts the ciphertext. The decrypted ciphertext shall be identical to the original plaintext.
- IUT generates multiple ciphertexts for the same plaintext, the same public key, and the same label, where the number of signatures is defined in section 4. All the generated ciphertexts shall be mutually different.

Test case 2 (Optional. Known Answer Test for ciphertexts)

- IUT shall generate correct ciphertext for the public key \((n, e)\), the plaintext, the label \(L\), the pseudorandom number generator function, the hash function, the mask generation function \(MGF\), and the seed given by JCATT.

Test case 3 (Optional. Known Answer Test with intermediate seed designated)

- IUT shall generate correct ciphertext for the public key \((n, e)\), the plaintext, the label \(L\), the hash function, the mask generation function \(MGF\) and the intermediate seed given by JCATT.

The hash function is selected from SHA-1, SHA-256, SHA-384, or SHA-512.
The mask generation function \(MGF\) is selected from the functions based on SHA-1, SHA-256, SHA-384, or SHA-512 as described in ANSI X9.44.
When the test case 2 is performed, the pseudorandom number generator function is selected from the algorithms listed in 3.1.4.2.
3.2.1.3 The Decryption Test

The following decryption tests are performed:

- The private key \((n, d)\) or \((p, q, d_P, d_Q, qI_m)\), the label \(L\), the hash function, the mask generation function \(MGF\), and the ciphertext are given to IUT. IUT decrypts the ciphertext and the resulting plaintext shall be identical to the original plaintext.

- The private key \((n, d)\) or \((p, q, d_P, d_Q, qI_m)\), the label \(L\), the hash function, the mask generation function \(MGF\), and the altered ciphertext are given to IUT. IUT shall detect the fact that the ciphertext has been altered.

The hash function is selected from SHA-1, SHA-256, SHA-384, or SHA-512. The mask generation function \(MGF\) is selected from the functions based on SHA-1, SHA-256, SHA-384, or SHA-512 as described in ANSI X9.44.
3.2.2 RSAES-PKCS1-v1.5

There are the following functions to be tested for RSAES-PKCS1-v1.5.

- Key Pair Generation
- Encryption
- Decryption

3.2.2.1 Key Pair Generation Test

The test is performed in the same way as the key pair generation test described in described in section 3.1.3.

3.2.2.2 The Encryption Test

For the encryption test, the following Test case 1, Test case 2, and Test case 3 are available. The Test case 1 is mandatory for validation and is performed by default.

**Test case 1** (The default and mandatory test)

- IUT encrypts the plaintext with the public key \((n, e)\) given by JCATT. JCATT then decrypts the ciphertext. The decrypted ciphertext shall be identical to the original plaintext.
- IUT shall never generate identical ciphertext when IUT encrypt the same plaintext with the same public key multiple times.

**Test case 2** (Optional. Known Answer Test for ciphertexts)

- IUT shall generate correct ciphertext for the public key \((n, e)\), the plaintext, the pseudorandom number generator function, and the seed given by JCATT.

**Test case 3** (Optional. Known Answer Test with intermediate PS designated)

- IUT shall generate correct ciphertext for the public key \((n, e)\), the plaintext, and the intermediate value \(PS\) given by JCATT.

When the test case 2 is performed, the pseudorandom number generator function is selected from the algorithms listed in 3.1.4.2.

3.2.2.3 The Decryption Test

The following decryption tests are performed:

- The private key \((n, d)\) or \((p, q, dP, dQ, qInv)\), and the ciphertext are given to IUT. IUT decrypts the ciphertext and the resulting plaintext shall be identical to the original plaintext.
- The private key \((n, d)\) or \((p, q, dP, dQ, qInv)\), and the altered ciphertext are given to IUT. IUT shall detect the fact that the ciphertext has been altered.
4 Conditions for Issuing Cryptographic Algorithm Validation Certificate

4.1 Details of Conditions

Requirements and default values of the parameters used for cryptographic algorithm implementation testing are shown in Table 4.1~Table 4.7. In these tables, the first columns indicate the functions to be tested. There are two classes of functions. One is the class of mandatory functions for validation. The other is the class of optional functions. Mandatory functions are indicated by highlighted text in these tables.

For Public-Key Cryptography, the conditions for issuing cryptographic algorithm validation certificate are:

- IUT shall implement at least one mandatory function.
- IUT shall pass the cryptographic algorithm implementation test.

<table>
<thead>
<tr>
<th>Function</th>
<th>Parameter</th>
<th>Default</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Domain Parameter Generation</strong></td>
<td>Bit length of $p$</td>
<td>1024</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>Bit length of SEED</td>
<td>160</td>
<td>Multiple of 8 and $160 \leq x \leq 16000$</td>
</tr>
<tr>
<td></td>
<td>The number of domain parameters to be generated</td>
<td>10</td>
<td>$\geq 10$</td>
</tr>
<tr>
<td><strong>Domain Parameter Validation</strong></td>
<td>Bit length of $p$</td>
<td>1024</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>Bit length of SEED</td>
<td>160</td>
<td>Multiple of 8 and $160 \leq x \leq 16000$</td>
</tr>
<tr>
<td></td>
<td>The number of domain parameters to be generated</td>
<td>10</td>
<td>$\geq 10$</td>
</tr>
<tr>
<td></td>
<td>The percentage of the data to be altered (%)</td>
<td>30</td>
<td>$1 \leq x \leq 99$</td>
</tr>
<tr>
<td><strong>Key Pair Generation</strong></td>
<td>Bit length of $p$</td>
<td>1024</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>The number of key pairs to be generated</td>
<td>10</td>
<td>$\geq 10$</td>
</tr>
<tr>
<td><strong>Signature Generation</strong></td>
<td>Bit length of $p$</td>
<td>1024</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>Bit length of plaintext</td>
<td>1024</td>
<td>Multiple of 8 and $\leq 16000$</td>
</tr>
<tr>
<td></td>
<td>The number of plaintexts</td>
<td>10</td>
<td>$\geq 10$</td>
</tr>
<tr>
<td></td>
<td>The number of signatures to test whether generated signatures vary.</td>
<td>10</td>
<td>$\geq 10$</td>
</tr>
<tr>
<td><strong>Signature Verification</strong></td>
<td>Bit length of $p$</td>
<td>1024</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>Bit length of plaintext</td>
<td>1024</td>
<td>Multiple of 8 and $\leq 16000$</td>
</tr>
<tr>
<td></td>
<td>The number of plaintexts</td>
<td>10</td>
<td>$\geq 10$</td>
</tr>
<tr>
<td></td>
<td>The percentage of the data to be altered (%)</td>
<td>30</td>
<td>$1 \leq x \leq 99$</td>
</tr>
</tbody>
</table>
Table 4.2: ECDSA over Prime Field

<table>
<thead>
<tr>
<th>Function</th>
<th>Parameter</th>
<th>Default</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain Parameter Generation</td>
<td>Bit length of $p$</td>
<td>160</td>
<td>Multiple of 8 and ≤ 16000</td>
</tr>
<tr>
<td></td>
<td>Bit length of $SEED$ to verify randomness of the curve</td>
<td>160</td>
<td>Multiple of 8 and ≤ 16000</td>
</tr>
<tr>
<td></td>
<td>The number of domain parameters to be generated</td>
<td>10</td>
<td>≥ 10</td>
</tr>
<tr>
<td>Domain Parameter Validation</td>
<td>None</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Key Pair Generation</td>
<td>The number of key pairs</td>
<td>10</td>
<td>≥ 10</td>
</tr>
<tr>
<td>Public Key Validation</td>
<td>The number of keys</td>
<td>10</td>
<td>≥ 10</td>
</tr>
<tr>
<td></td>
<td>The percentage of the data to be altered (%)</td>
<td>30</td>
<td>$1 \leq x \leq 99$</td>
</tr>
<tr>
<td>Signature Generation</td>
<td>Hash Function</td>
<td>SHA-256</td>
<td>SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, or RIPEMD-160</td>
</tr>
<tr>
<td></td>
<td>Bit length of plaintext</td>
<td>1024</td>
<td>Multiple of 8 and ≤ 16000</td>
</tr>
<tr>
<td></td>
<td>The number of plaintexts</td>
<td>10</td>
<td>≥ 10</td>
</tr>
<tr>
<td></td>
<td>The number of signatures to test whether generated signatures vary.</td>
<td>10</td>
<td>≥ 10</td>
</tr>
<tr>
<td>Signature Verification</td>
<td>Hash Function</td>
<td>SHA-256</td>
<td>SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, or RIPEMD-160</td>
</tr>
<tr>
<td></td>
<td>Bit length of plaintext</td>
<td>1024</td>
<td>Multiple of 8 and ≤ 16000</td>
</tr>
<tr>
<td></td>
<td>The number of plaintexts</td>
<td>10</td>
<td>≥ 10</td>
</tr>
<tr>
<td></td>
<td>The percentage of the data to be altered (%)</td>
<td>30</td>
<td>$1 \leq x \leq 99$</td>
</tr>
</tbody>
</table>
Table 4.3: ECDSA over Binary Field

<table>
<thead>
<tr>
<th>Function</th>
<th>Parameter</th>
<th>Default</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Domain Parameter Generation</strong></td>
<td>Degree of Irreducible Polynomial</td>
<td>163</td>
<td>( \leq 16000 )</td>
</tr>
<tr>
<td></td>
<td>Bit length of SEED to verify randomness of the curve</td>
<td>160</td>
<td>Multiple of 8 and ( \leq 16000 )</td>
</tr>
<tr>
<td></td>
<td>The number of domain parameters to be generated</td>
<td>10</td>
<td>( \geq 10 )</td>
</tr>
<tr>
<td><strong>Domain Parameter Validation</strong></td>
<td>None</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td><strong>Key Pair Generation</strong></td>
<td>The number of key pairs</td>
<td>10</td>
<td>( \geq 10 )</td>
</tr>
<tr>
<td><strong>Public Key Validation</strong></td>
<td>The number of keys</td>
<td>10</td>
<td>( \geq 10 )</td>
</tr>
<tr>
<td></td>
<td>The percentage of the data to be altered (%)</td>
<td>30</td>
<td>( 1 \leq x \leq 99 )</td>
</tr>
<tr>
<td><strong>Signature Generation</strong></td>
<td>Hash Function</td>
<td>SHA-256</td>
<td>SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, or RIPEMD-160</td>
</tr>
<tr>
<td></td>
<td>Bit length of plaintext</td>
<td>1024</td>
<td>Multiple of 8 and ( \leq 16000 )</td>
</tr>
<tr>
<td></td>
<td>The number of plaintexts</td>
<td>10</td>
<td>( \geq 10 )</td>
</tr>
<tr>
<td></td>
<td>The number of signatures to test whether generated signatures vary.</td>
<td>10</td>
<td>( \geq 10 )</td>
</tr>
<tr>
<td><strong>Signature Verification</strong></td>
<td>Hash Function</td>
<td>SHA-256</td>
<td>SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, or RIPEMD-160</td>
</tr>
<tr>
<td></td>
<td>Bit length of plaintext</td>
<td>1024</td>
<td>Multiple of 8 and ( \leq 16000 )</td>
</tr>
<tr>
<td></td>
<td>The number of plaintexts</td>
<td>10</td>
<td>( \geq 10 )</td>
</tr>
<tr>
<td></td>
<td>The percentage of altered data (%)</td>
<td>30</td>
<td>( 1 \leq x \leq 99 )</td>
</tr>
</tbody>
</table>

Table 4.4: RSASSA-PKCS1-v1_5

<table>
<thead>
<tr>
<th>Function</th>
<th>Parameter</th>
<th>Default</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Key Pair Generation</strong></td>
<td>Bit length of key</td>
<td>1024</td>
<td>( 1024 \leq x \leq 16000 )</td>
</tr>
<tr>
<td></td>
<td>Type of private keys with CRT</td>
<td>with CRT or without CRT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Type of public keys ( e )</td>
<td>65537</td>
<td>65537 or Random</td>
</tr>
<tr>
<td></td>
<td>The number of key pairs</td>
<td>10</td>
<td>( \geq 10 )</td>
</tr>
<tr>
<td><strong>Signature Generation</strong></td>
<td>Bit length of key</td>
<td>1024</td>
<td>( 1024 \leq x \leq 16000 )</td>
</tr>
<tr>
<td></td>
<td>Type of private keys with CRT</td>
<td>with CRT or without CRT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Hash Function</td>
<td>SHA-256</td>
<td>SHA-1, SHA-256, SHA-384, or SHA-512</td>
</tr>
<tr>
<td></td>
<td>Bit length of plaintext</td>
<td>1024</td>
<td>Multiple of 8 and ( \leq 16000 )</td>
</tr>
<tr>
<td></td>
<td>The number of plaintexts</td>
<td>10</td>
<td>( \geq 10 )</td>
</tr>
<tr>
<td><strong>Signature Verification</strong></td>
<td>Bit length of key</td>
<td>1024</td>
<td>( 1024 \leq x \leq 16000 )</td>
</tr>
<tr>
<td></td>
<td>Type of public keys ( e )</td>
<td>65537</td>
<td>65537 or Random</td>
</tr>
<tr>
<td></td>
<td>Hash Function</td>
<td>SHA-256</td>
<td>SHA-1, SHA-256, SHA-384, or SHA-512</td>
</tr>
<tr>
<td></td>
<td>Bit length of plaintext</td>
<td>1024</td>
<td>Multiple of 8 and ( \leq 16000 )</td>
</tr>
<tr>
<td></td>
<td>The number of plaintexts</td>
<td>10</td>
<td>( \geq 10 )</td>
</tr>
<tr>
<td></td>
<td>The percentage of altered data (%)</td>
<td>30</td>
<td>( 1 \leq x \leq 99 )</td>
</tr>
</tbody>
</table>

17/21
<table>
<thead>
<tr>
<th>Function</th>
<th>Parameter</th>
<th>Default</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Key Pair Generation</strong></td>
<td>Bit length of key</td>
<td>1024</td>
<td>$1024 \leq x \leq 16000$</td>
</tr>
<tr>
<td></td>
<td>Type of private keys</td>
<td>with CRT</td>
<td>with CRT or without CRT</td>
</tr>
<tr>
<td></td>
<td>Type of public keys $e$</td>
<td>65537</td>
<td>65537 or Random</td>
</tr>
<tr>
<td></td>
<td>The number of key pairs</td>
<td>10</td>
<td>$\geq 10$</td>
</tr>
<tr>
<td><strong>Signature Generation</strong></td>
<td>Bit length of key</td>
<td>1024</td>
<td>$1024 \leq x \leq 16000$</td>
</tr>
<tr>
<td></td>
<td>Type of private keys</td>
<td>with CRT</td>
<td>with CRT or without CRT</td>
</tr>
<tr>
<td></td>
<td>Hash Function</td>
<td>SHA-256</td>
<td>SHA-1, SHA-256, SHA-384, or SHA-512</td>
</tr>
<tr>
<td></td>
<td>Mask generation function $MGF$</td>
<td>ANSI X9.44</td>
<td>ANSI X9.44 SHA-1, SHA-256, SHA-384, or SHA-512</td>
</tr>
<tr>
<td></td>
<td>Bit length of $salt$</td>
<td>256</td>
<td>Multiple of 8 and $\leq 16000$</td>
</tr>
<tr>
<td></td>
<td>Bit length of plaintext</td>
<td>1024</td>
<td>Multiple of 8 and $\leq 16000$</td>
</tr>
<tr>
<td></td>
<td>The number of plaintexts</td>
<td>10</td>
<td>$\geq 10$</td>
</tr>
<tr>
<td></td>
<td>The number of signatures to test whether generated signatures vary.</td>
<td>10</td>
<td>$\geq 10$</td>
</tr>
<tr>
<td></td>
<td>PRNG (For the test case 2)</td>
<td>-</td>
<td>Refer to section 3.1.4.2</td>
</tr>
<tr>
<td><strong>Signature Verification</strong></td>
<td>Bit length of key</td>
<td>1024</td>
<td>$1024 \leq x \leq 16000$</td>
</tr>
<tr>
<td></td>
<td>Type of public keys $e$</td>
<td>65537</td>
<td>65537 or Random</td>
</tr>
<tr>
<td></td>
<td>Hash Function</td>
<td>SHA-256</td>
<td>SHA-1, SHA-256, SHA-384, or SHA-512</td>
</tr>
<tr>
<td></td>
<td>Mask generation function $MGF$</td>
<td>ANSI X9.44</td>
<td>ANSI X9.44 SHA-1, SHA-256, SHA-384, or SHA-512</td>
</tr>
<tr>
<td></td>
<td>Bit length of $salt$</td>
<td>256</td>
<td>Multiple of 8 and $\leq 16000$</td>
</tr>
<tr>
<td></td>
<td>Bit length of plaintext</td>
<td>1024</td>
<td>Multiple of 8 and $\leq 16000$</td>
</tr>
<tr>
<td></td>
<td>The number of plaintexts</td>
<td>10</td>
<td>$\geq 10$</td>
</tr>
<tr>
<td></td>
<td>The percentage of altered data (%)</td>
<td>30</td>
<td>$1 \leq x \leq 99$</td>
</tr>
</tbody>
</table>
### Table 4.6: RSA-OAEP

<table>
<thead>
<tr>
<th>Function</th>
<th>Parameter</th>
<th>Default</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bit length of key</td>
<td>1024</td>
<td>$1024 \leq x \leq 16000$</td>
</tr>
<tr>
<td></td>
<td>Type of private keys</td>
<td>with CRT</td>
<td>with CRT or without CRT</td>
</tr>
<tr>
<td></td>
<td>Type of public keys $e$</td>
<td>65537</td>
<td>65537 or Random</td>
</tr>
<tr>
<td></td>
<td>The number of key pairs</td>
<td>10</td>
<td>$\geq 10$</td>
</tr>
<tr>
<td><strong>Key Pair Generation</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bit length of key</td>
<td>1024</td>
<td>$1024 \leq x \leq 16000$</td>
</tr>
<tr>
<td></td>
<td>Type of public keys $e$</td>
<td>65537</td>
<td>65537 or Random</td>
</tr>
<tr>
<td></td>
<td>Hash Function</td>
<td>SHA-256</td>
<td>SHA-1, SHA-256, SHA-384, or SHA-512</td>
</tr>
<tr>
<td></td>
<td>Mask generation function $MGF$</td>
<td>ANSI X9.44 SHA-256</td>
<td>ANSI X9.44 SHA-1, SHA-256, SHA-384, or SHA-512</td>
</tr>
<tr>
<td></td>
<td>Bit length of label $L$</td>
<td>0</td>
<td>Multiple of 8 and $\leq 16000$</td>
</tr>
<tr>
<td></td>
<td>Bit length of plaintext</td>
<td>64</td>
<td>Multiple of 8 and $\leq 16000$</td>
</tr>
<tr>
<td></td>
<td>The number of plaintexts</td>
<td>10</td>
<td>$\geq 10$</td>
</tr>
<tr>
<td></td>
<td>The number of ciphertexts</td>
<td>10</td>
<td>$\geq 10$</td>
</tr>
<tr>
<td></td>
<td>PRNG (For the test case 2)</td>
<td>-</td>
<td>Refer to section 3.1.4.2</td>
</tr>
<tr>
<td><strong>Encryption</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bit length of key</td>
<td>1024</td>
<td>$1024 \leq x \leq 16000$</td>
</tr>
<tr>
<td></td>
<td>Type of private keys</td>
<td>with CRT</td>
<td>with CRT or without CRT</td>
</tr>
<tr>
<td></td>
<td>Hash Function</td>
<td>SHA-256</td>
<td>SHA-1, SHA-256, SHA-384, or SHA-512</td>
</tr>
<tr>
<td></td>
<td>Mask generation function $MGF$</td>
<td>ANSI X9.44 SHA-256</td>
<td>ANSI X9.44 SHA-1, SHA-256, SHA-384, or SHA-512</td>
</tr>
<tr>
<td></td>
<td>Bit length of label $L$</td>
<td>256</td>
<td>Multiple of 8 and $\leq 16000$</td>
</tr>
<tr>
<td></td>
<td>Bit length of plaintext</td>
<td>64</td>
<td>Multiple of 8 and $\leq 16000$</td>
</tr>
<tr>
<td></td>
<td>The number of ciphertexts</td>
<td>10</td>
<td>$\geq 10$</td>
</tr>
<tr>
<td></td>
<td>The percentage of altered data (%)</td>
<td>30</td>
<td>$1 \leq x \leq 99$</td>
</tr>
</tbody>
</table>

### Table 4.7: RSAES-PKCS1-v1_5

<table>
<thead>
<tr>
<th>Function</th>
<th>Parameter</th>
<th>Default</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bit length of key</td>
<td>1024</td>
<td>$1024 \leq x \leq 16000$</td>
</tr>
<tr>
<td></td>
<td>Type of private keys</td>
<td>with CRT</td>
<td>with CRT or without CRT</td>
</tr>
<tr>
<td></td>
<td>Type of public keys $e$</td>
<td>65537</td>
<td>65537 or Random</td>
</tr>
<tr>
<td></td>
<td>The number of key pairs</td>
<td>10</td>
<td>$\geq 10$</td>
</tr>
<tr>
<td><strong>Key Pair Generation</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bit length of key</td>
<td>1024</td>
<td>$1024 \leq x \leq 16000$</td>
</tr>
<tr>
<td></td>
<td>Type of public keys $e$</td>
<td>65537</td>
<td>65537 or Random</td>
</tr>
<tr>
<td></td>
<td>PRNG (For the test case 2)</td>
<td>-</td>
<td>Refer to section 3.1.4.2</td>
</tr>
<tr>
<td><strong>Encryption</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bit length of key</td>
<td>1024</td>
<td>$1024 \leq x \leq 16000$</td>
</tr>
<tr>
<td></td>
<td>Type of public keys $e$</td>
<td>65537</td>
<td>65537 or Random</td>
</tr>
<tr>
<td></td>
<td>Bit length of plaintext</td>
<td>64</td>
<td>Multiple of 8 and $\leq 16000$</td>
</tr>
<tr>
<td></td>
<td>The number of plaintexts</td>
<td>10</td>
<td>$\geq 10$</td>
</tr>
<tr>
<td></td>
<td>The number of ciphertexts</td>
<td>10</td>
<td>$\geq 10$</td>
</tr>
<tr>
<td></td>
<td>The percentage of altered data (%)</td>
<td>30</td>
<td>$1 \leq x \leq 99$</td>
</tr>
</tbody>
</table>

19/21
Supplementary Provision
This procedure shall come into force as of April 1, 2009, and shall be applicable as of April 1, 2009.

Supplementary Provision
This procedure shall come into force as of September 30, 2009, and shall be applicable as of September 30, 2009.

References


<table>
<thead>
<tr>
<th>Date of Revision</th>
<th>Prepared / Approved by</th>
<th>Revision Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>April 1, 2009</td>
<td>Hashimoto / Nakata</td>
<td>Newly Created</td>
</tr>
<tr>
<td>September 30, 2009</td>
<td>Hashimoto / Nakata</td>
<td>Partially Revised</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Described selectable PRNGs.)</td>
</tr>
</tbody>
</table>