# Specification of ESIGN Identification

## 1 Introduction

We describe the specification of an identification scheme based on ESIGN signature scheme. The identification scheme, ESIGN Identification, is converted in a traditional way from ESIGN signature scheme. ESIGN is originally a digital signature scheme specified by a triplet of primitive algorithms, $(\mathcal{G}, \mathcal{S}, \mathcal{V})$, along with a hash function, where $\mathcal{G}$ is called the key generation algorithm, $\mathcal{S}$ the signing algorithm, and $\mathcal{V}$ the verifying algorithm. We will describe the specifications of these algorithms in Sec. 4. We also use some auxiliary algorithms, such as a pseudo-random number generator, a primality test algorithm, and a hash function, which we mention in Sec. 5 and Sec. 9 later.

## 2 Criteria of Design

### 2.1 Design Policy

Security in a cryptosystem is clearly the most important criterion: We make in a traditional way an identification scheme from ESIGN signature scheme. ESIGN as a signature scheme can be proven secure in the strongest security notion (existentially unforgeable against adaptively-chosen message attacks) in the random oracle model. The identification scheme obtained by this way then can be proven secure for an identification scheme in the RO model (See the document of "Self-Evaluation of ESIGN Identification"). The definition of security for an identification scheme follows that in [12].

Efficiency is also a very important factor in a cryptosystem – performance and amount of resource when implemented in software/hardware. ESIGN identification is at least as efficient as any well-known identification scheme (See 2.2).

### 2.2 ESIGN Signature

ESIGN is a digital signature scheme that can be proven secure in the strongest security notion (existentially unforgeable against adaptively-chosen message attacks) in the random oracle model.

To achieve the security, we adopt the random oracle paradigm along with a reasonable intractable assumption. In the random oracle paradigm, security of a cryptosystem is proved assuming hash functions are modeled as random oracles. This paradigm was originally proposed by Bellare and Rogaway in [3], and is rapidly becoming a standard approach to achieve a provably-secure cryptosystem. Security of ESIGN, in the random oracle model, can be assured under an intractable assumption, which we name the approximate $e$-th root assumption. This assumption is an approximate version of RSA assumption.

As for efficiency, signature generation with ESIGN is ten times more efficient than that achieved with RSA-based signature schemes, while their verification performances are comparable. Compared to EC(Elliptic Curve)-based signature schemes, ESIGN is several times faster in terms of signature and verification performance.

## 3 Notations

- $a := b$: the value of $b$ is substituted for $a$, or $a$ is defined as $b$.
- $\mathbb{Z}$ : the set of integers.
- $\mathbb{Z}/n\mathbb{Z} := \{0, 1, \ldots, n-1\}$.
- Let $A$, $B$ be sets. $A \backslash B := \{x \mid x \in A \wedge x \notin B\}$.
- Let $A$ be a set. For $k \in \mathbb{N}$, $A^k$: the set of all $k$-tuples of elements in $A$ (i.e., $A^k := \underbrace{A \times \cdots \times A}_{k}$).
- $(\mathbb{Z}/n\mathbb{Z})^{\times} := \{1, 2, \ldots, n-1\} \backslash \{x \mid \gcd(x, n) \neq 1\}$.
- $\{0,1\}^*$ is the set of finite strings. $\{0,1\}^*$ is also denoted by $\mathbf{B}$.
- $\{0,1\}^i$ is the set of $i$ bit length bit strings. $\{0,1\}^i$ is also denoted by $\mathbf{B}_i$.

- Let $a \in \mathbb{Z}$. $\mathbf{B}_i[a]$ denotes a bit string $(a_{i-1}, a_{i-2}, \ldots, a_0) \in \mathbf{B}_i$ such that
$$a = a_0 + 2a_1 + 2^2 a_2 + \cdots 2^{i-1} a_{i-1}.$$
- Let $a := (a_{i-1}, a_{i-2}, \ldots, a_0) \in \mathbf{B}_i$. $\mathbf{I}[a]$ denotes an integer $b \in \mathbb{Z}$ such that
$$b = a_0 + 2a_1 + 2^2 a_2 + \cdots 2^{i-1} a_{i-1}.$$
- If $a \in \mathbf{B}_i$, $|a| := i$.
- $a \equiv b \pmod{n}$ means $a - b$ is divided by $n$. $a := b \bmod n$ denotes $a \in \mathbb{Z}/n\mathbb{Z}$ and $a \equiv b \pmod{n}$.
- Let $a \in \mathbf{B}$ and $b \in \mathbf{B}$. $a||b$ denotes the concatenation of $a$ and $b$. For example, $(0,1,0,0)||(1,1,0) = (0,1,0,0,1,1,0)$.
- Let $a \in \mathbf{B}$. $a^k := \underbrace{a||\cdots||a}_{k}$.
- Let $X \in \mathbf{B}$. $[X]^{pLen}$ denotes the most $pLen$ significant bits of $X$.
- Let $a \in \mathbf{B}_i$ and $b \in \mathbf{B}_i$. $a \oplus b$ means the bit-wise exclusive-or operation. (i.e., $a \oplus b \in \mathbf{B}_i$.)

# 4 Cryptographic Primitives

ESIGN is specified by a triplet of primitive algorithms, $(\mathcal{G}, \mathcal{S}, \mathcal{V})$, where $\mathcal{G}$ is called the key generation algorithm, $\mathcal{S}$ the signing algorithm, and $\mathcal{V}$ the verifying algorithm.

If a variable, $x$, in an input or output in this specification is in $\mathbb{Z}$, then it should be in the binary form, $\mathbf{B}_i[x]$, where $i$ is an arbitrary length (specified by the interface with an application/protocol) with $x < 2^i$.

## 4.1 Key Generation: $\mathcal{G}$

The input and output of $\mathcal{G}$ are as follows:

**[Input ]** Security parameter $k(= pLen) \in \mathbb{Z}$.

**[Output ]** The pair of public-key, $(n, e, pLen) \in \mathbb{Z}^3$, and the secret-key, $(p, q) \in \mathbb{Z}^2$.

The operation of $\mathcal{G}$, on input $k$ is as follows:

- Choose two distinct primes, $p$, $q$, of size $k$ and compute $n := p^2q$.
- Select an integer $e > 4$.
- Set $pLen := k$.
- Output the binary coding of $(n, e, pLen)$ and $(p, q)$.

### 4.2 Signature Generation: $\mathcal{S}$

The input and output of $\mathcal{S}$ are as follows:

**[Input ]** A string, $m \in \{0,1\}^{pLen-1}$ along with (the binary coding of) a public-key, $(n, e, pLen)$.

**[Output ]** A binary string, $s \in \{0,1\}^{3pLen}$.

The operation of $\mathcal{S}$, on input $m$, $(p, q)$, and $(n, e, pLen)$, is as follows:

1. Pick $r$ at random and uniformly from $(\mathbb{Z}/pq\mathbb{Z})\backslash p\mathbb{Z} := \{r \in \mathbb{Z}/pq\mathbb{Z} | \gcd(r, p) = 1\}$.
2. Set $z := (0||m||0^{2 \cdot pLen})$ and $\alpha := (I(z) - r^e) \bmod n$.
3. Set $(w_0, w_1)$ such that
$$\begin{aligned} w_0 &:= \lceil \frac{\alpha}{pq} \rceil, \\ w_1 &:= w_0 \cdot pq - \alpha. \end{aligned}$$
4. If $w_1 \geq 2^{2pLen-1}$, then go back to Step 1. (That is, if the most significant bit of $w_1$ is 1, then go back to Step 1.)
5. Set $t := \frac{w_0}{er^{e-1}} \bmod p$ and $s := B_{3pLen}[(r + tpq) \bmod n]$.
6. Output $s$.

### 4.3 Signature Verification: $\mathcal{V}$

The input and output of $\mathcal{V}$ are as follows:

**[Input ]** The pair of strings, $(m, s)$, along with (the binary coding of) the public-key, $(n, e, pLen)$.

**[Output ]** A bit — '1' represents *valid* and '0' represents *invalid*).

The operation of $\mathcal{V}$, on input $(m, s)$ along with $(n, e, pLen)$ is as follows:

- Check whether the following equation holds or not:

$$[B_{3pLen}[I(s)^e \bmod n]]^{pLen} = 0||m.$$

- If it holds, output '1' (rep.*valid*), otherwise output '0' (rep. *invalid*).

# 5 Auxiliary Algorithms

Here we describe the auxiliary algorithms used in this paper.

- **[PRNG]** In the key-generation and signature-generation algorithms, a pseudo-random number generator (PRNG) is used to pick up a random number from an appropriate domain. For a practical construction of PRNGs, the reader is referred to [14, Annex D.6] or [17, Chapter 6].
- **[Primality Test]** In the key-generation and signature-generation algorithms, a primality testing algorithm is used to pick up a prime number with appropriate bit-length. A practical construction of a primality testing algorithm is, for instance, Miller-Rabin Test [14, Annex A.15.1].
- **[Hash Function]** A construction of a hash function is described in Sec. 9, which is used in the signature-generation and verification algorithms.
- **[Basic Operations]** Basic operations over groups, rings, and fields, like multiplication, addition, etc., follow algorithms in [14, Annex A.1-3].

# 6 Specification of ESIGN Identification

This section details the procedure of ESIGN identification.

**[Set-UP]** User $A$ generates a public key, $(n, e, \mathrm{HID}, pLen)$, and the corresponding secret key, $(p, q)$, following the procedure of key generator $\mathcal{G}$. $A$ then register the public key to the server(or verifier) $B$.

**[Identification]**

1. Verifier $B$ generates a random string $r \in \{0,1\}^{mLen}$ and send them to user $A$.
2. $A$ makes a ESIGN signature, $s$, on random string $r$, using the secret key $(p,q)$, and return the signature to $B$.
3. $B$ verifies the validity of the signature: Check whether the following equation holds or not,

$$[B_{3pLen}[I(s)^e \bmod n]]^{pLen} = 0||H(m).$$

# 7 Specification of ESIGN Signature

Here we describe the specification of ESIGN.

If a variable, $x$, in an input or output in this specification is in $\mathbb{Z}$, then it should be in the binary form, $\mathbf{B}_i[x]$, where $i$ is an arbitrary length (specified by the interface with an application/protocol) with $x < 2^i$.

## 7.1 Key Generation: $\mathcal{G}$

The input and output of $\mathcal{G}$ are as follows:

**[Input ]** Security parameter $k(= pLen) \in \mathbb{Z}$.

**[Output ]** The pair of public-key, $(n, e, pLen, \mathrm{HID}) \in \mathbb{Z}^4$, and the secret-key, $(p,q) \in \mathbb{Z}^2$.

The operation of $\mathcal{G}$, on input $k$ is as follows:

- Choose two distinct primes, $p$, $q$, of size $k$ and compute $n := p^2 q$.
- Select an integer $e > 4$.
- Set $pLen := k$.
- Pick up HID where HID indicates the identity of a (hash) function $H : \{0,1\}^* \rightarrow \{0,1\}^{pLen-1}$ in the pre-prepared hash function list.
- Output the binary coding of $(n, e, pLen, \mathrm{HID})$ and $(p,q)$.

**Remark:**

Since $0||H(x)$, not $H(x)$, is always required in the signing and verification procedures, $H(x)$ can be realized by using hash function $H' : \{0,1\}^* \longrightarrow \{0,1\}^{pLen}$ as follows: first $H'(x)$ is computed, and the most significant bit of $H'(x)$ is set to '0' while preserving the other bits. The resulting value is $0||H(x)$.

## 7.2 Signature Generation: $\mathcal{S}$

The input and output of $\mathcal{S}$ are as follows:

**[Input ]** A message, $m \in \{0,1\}^*$ along with (the binary coding of) a public-key, $(n, e, pLen, \mathrm{HID})$.

**[Output ]** A binary string, $s \in \{0,1\}^{3pLen}$.

The operation of $\mathcal{S}$, on input $m$, $(p,q)$, and $(n, e, pLen, \mathrm{HID})$, is as follows:

1. Pick $r$ at random and uniformly from $(\mathbb{Z}/pq\mathbb{Z})\backslash p\mathbb{Z} := \{r \in \mathbb{Z}/pq\mathbb{Z} | \gcd(r,p) = 1\}$.
2. Set $z := (0||H(m)||0^{2 \cdot pLen})$ and $\alpha := (I(z) - r^e) \bmod n$.
3. Set $(w_0, w_1)$ such that
$$\begin{aligned} w_0 &:= \lceil \frac{\alpha}{pq} \rceil, \\ w_1 &:= w_0 \cdot pq - \alpha. \end{aligned}$$
4. If $w_1 \geq 2^{2pLen-1}$, then go back to Step 1. (That is, if the most significant bit of $w_1$ is 1, then go back to Step 1.)
5. Set $t := \frac{w_0}{e r^{e-1}} \bmod p$ and $s := B_{3pLen}[(r + tpq) \bmod n]$.
6. Output $s$.

## 7.3 Signature Verification: $\mathcal{V}$

The input and output of $\mathcal{V}$ are as follows:

**[Input ]** The pair of message and signature, $(m, s)$, along with (the binary coding of) the public-key, $(n, e, pLen, \mathrm{HID})$.

**[Output ]** A bit — '1' represents *valid* and '0' represents *invalid*).

The operation of $\mathcal{V}$, on input $(m,s)$ along with $(n,e,pLen,\mathrm{HID})$ is as follows:

- Check whether the following equation holds or not:

$$[B_{3pLen}[I(s)^e \bmod n]]^{pLen} = 0||H(m).$$

- If it holds, output '1' (rep.*valid*), otherwise output '0' (rep. *invalid*).

# 8 Recommended Parameters

We recommend ESIGN parameters as follows:

- $k$: more than or equal to 320 (the size of $n$ should be more than 960 bits), and
- $e$: more than or equal to 8.

We used 1152 bits as the size of $n$ and $e = 32$ in Sec. 4 in the document "Self-Evaluation of ESIGN Identification".

# 9 Hash Function

In the key-generation algorithm, a hash function used in the signature-generation and verification algorithms is picked up from the pre-prepared hash function list. ESIGN can be proven secure if the hash function in it is modeled as a random oracle.

We show a typical construction of a hash function with $pLen > 160$ out of SHA (NIST Secure Hash Algorithm), which was suggested by Bellare and Rogaway [4].

We denote by $\mathrm{SHA}_\sigma(x)$ the 160-bit result of SHA applied to $x$, except that the 160-bit "starting value" in the algorithm description is taken to be $ABCDE = \sigma$. Let $\mathrm{SHA}^l_\sigma(x)$ denote the first $l$-bits of $\mathrm{SHA}_\sigma(x)$. Fix the notation $< i >$ for $i$ encoded as a binary 32-bit word. We define function $H$ as:

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

where $l = \lfloor \frac{3k}{80} \rfloor$, and $L_l = pLen - 80l$.

## References


[1] Abdalla, M., Bellare, M. and Rogaway, P.: DHES: An Encryption Scheme Based on the Diffie-Hellman Problem, Submission to IEEE P1363a (1998, August).

[2] Adleman, L.M. and McCurley, K.S.: Open Problems in Number Theoretic Complexity,II (open problems: C7, O7a and O7b), Proc. of ANTS-I, LNCS 877, Springer-Verlag, pp.291-322 (1995).

[3] 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).

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

[5] 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).

[6] Boneh, D., Durfee, G. and Howgrave-Graham, N.: Factoring $N = p^r q$ for Large $r$, Proc. of Crypto'99, LNCS 1666, Springer-Verlag, pp.326-337 (1999)

[7] Brickell, E. and DeLaurentis, J.: An Attack on a Signature Scheme Proposed by Okamoto and Shiraishi, Proc. of Crypto'85, LNCS 218, Springer-Verlag, pp.28-32 (1986)

[8] Brickell, E. and Odlyzko: Cryptanalysis: A Survey of Recent Results, Chap.10, Contemporary Cryptology, Simmons (Ed.), IEEE Press, pp.501–540 (1991).

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

[10] Cryptography Using Compaq MultiPrime Technology in a Parallel Processing Environment, Enterprise Security Solutions, Electronic Commerce Technical Brief, Compaq Computer Corporation, http://www6.compaq.com/solutions/security/ (2000)

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

[12] Feige, U., Fiat, A. and Shamir, A.: Zero-Knowledge Proofs of Identity, Journal of Cryptology, 1, 2, pp.77-94 (1988)

[13] Girault, M., Toffin, P. and Vallée, B.: Computation of Approximate $L$-th Roots Modulo $n$ and Application to Cryptography, Proc. of Crypto'88, LNCS 403, Springer-Verlag, pp.100-117 (1990)

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

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

[16] 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.

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

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

[19] Okamoto, T., Fujisaki, E. and Morita, H.: TSH-ESIGN: Efficient Digital Signature Scheme Using Trisection Size Hash, submission to P1363a (1998).

[20] Okamoto, T. and Shiraishi, A.: A Fast Signature Scheme Based on Quadratic Inequalities, Proc. of the ACM Symposium on Security and Privacy, ACM Press (1985).

[21] Peralta, R.: Bleichenbacher's improvement for factoring numbers of the form $N = PQ^2$ (private communication) (1997).

[22] Peralta, R. and Okamoto, E.: Faster Factoring of Integers of a Special Form, IEICE Trans. Fundamentals, E79-A, 4, pp.489-493 (1996).

[23] Pollard, J.L.: Manuscript (1997).

[24] Silverman, R.D.,: A Cost-Based Security Analysis of Symmetric and Asymmetric Key Lengths, Bulletin number13, RSA Laboratories, April 2000.

[25] 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.

[26] Vallée, B., Girault, M. and Toffin, P.: How to Break Okamoto's Cryptosystem by Reducing Lattice Bases, Proc. of Eurocrypt'88, LNCS 330, Springer-Verlag, pp.281-291 (1988)

[27] Vallée, B., Girault, M. and Toffin, P.: How to Guess $L$-th Roots Modulo $n$ by Reducing Lattice Bases, Proc. of Conference of ISSAC-88 and AAECC-6, (1988)