

TSH-ESIGN: Efficient Digital Signature Scheme
Using Trisection Size Hash
(Submission to P1363a)

Tatsuaki Okamoto Eiichiro Fujisaki Hikaru Morita

NTT Laboratories
1-1 Hikarino-oka, Yokosuka-shi, 239-0847 Japan
Email: {okamoto, fujisaki, morita}@sucaba.isl.ntt.co.jp

November 1998

Abstract

We describe a practical digital signature scheme, TSH-ESIGN, which is more efficient than any representative signature scheme such as elliptic curve and RSA based signature schemes. Moreover, TSH-ESIGN is provably secure in the strongest sense (existentially unforgeable against adaptive chosen message attacks) in the random oracle model under the approximate e -th root assumption which is an approximate version of the RSA assumption.

The digital signature scheme described in this contribution is obtained by combining two results: one [5] is by Okamoto, and the other [4] by Fujisaki and Okamoto.

Contents

1	Description of TSH-ESIGN	3
1.1	Overview	3
1.2	Key Generation: \mathcal{G}	3
1.3	Signature Generation: \mathcal{S}	3
1.4	Signature Verification: \mathcal{V}	4
2	Attributes and Advantages of TSH-ESIGN	4
3	Security Assessment of TSH-ESIGN	5
4	Limitations	5
5	Intellectual Property Statement	6
A	Appendix	7

1 Description of TSH-ESIGN

1.1 Overview

This section describes the proposed digital signature scheme, TSH-ESIGN, which is specified by triplet $(\mathcal{G}, \mathcal{S}, \mathcal{V})$, where \mathcal{G} is the key generation operation, \mathcal{S} signing operation, and \mathcal{V} verifying operation.

1.2 Key Generation: \mathcal{G}

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

[**Input**] Security parameter $k(= pLen)$, which is a positive integer.

[**Output**] A pair of public-key, $(n, e, H, pLen)$, and secret-key, (p, q) .

The operation of \mathcal{G} , on input 1^k , is as follows:

- Choose two primes p, q ($|p| = |q| = k$), and compute $n := p^2q$. Here, $p - 1 = p'u$ and $q - 1 = q'v$ such that p' and q' are primes, and $|u|$ and $|v|$ are $O(\log k)$.
- Select an integer $e > 4$ (e.g., $e = 2^l, l = O(\log n)$).
- Set $pLen := k$.
- Select a (hash) function $H: \{0, 1\}^* \rightarrow \{0, 1\}^{pLen-1}$.

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

1.3 Signature Generation: \mathcal{S}

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

[**Input**] Message $M \in \{0, 1\}^{mLen}$ along with public-key $(n, e, H, pLen)$.

[**Output**] Signature S .

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

1. Pick r at random and uniformly from $(\mathbb{Z}/pq\mathbb{Z}) \setminus p\mathbb{Z} := \{r \in \mathbb{Z}/pq\mathbb{Z} \mid \gcd(r, p) = 1\}$.
2. Set $z \leftarrow (0||H(M)||0^{2pLen})$ and $\alpha \leftarrow (z - r^e) \bmod n$.
3. Set (w_0, w_1) such that

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

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

If $w_1 \geq 2^{2pLen-1}$, then go back to Step 1. (I.e., if the most significant bit of w_1 is 1, then go back to Step 1.)

4. Set $t \leftarrow \frac{w_0}{er^{e-1}} \bmod p$, and $S \leftarrow (r + tpq) \bmod n$.
5. Output S as the signature of M .

1.4 Signature Verification: \mathcal{V}

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

[Input] Pair of signature S and message M along with public-key $(n, e, H, pLen)$.

[Output] Verification result (*valid* or *invalid*).

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

- Check whether the following equation hold or not:

$$[S^e \bmod n]^{pLen} == 0||H(M). \quad (3)$$

Here $[X]^k$ denotes the most significant k bits of X .

- If it holds, output *valid* (or 1). Otherwise output *invalid* (or 0).

2 Attributes and Advantages of TSH-ESIGN

The advantage of TSH-ESIGN is its efficiency and provable security. As for efficiency, TSH-ESIGN is more efficient than any representative signature schemes such as elliptic curve and RSA based signature schemes.

As for security, TSH-ESIGN can be proven to be secure in the strongest sense (existentially unforgeable against adaptive chosen message attacks: EUF-CMA) in the random oracle model under an reasonable assumption, the approximate e -th root (AERP) assumption, which is an approximate version of the RSA assumption. (We will describe the security in more detail in the next section.) Here, we show a table of comparing the security of TSH-ESIGN, RSA based scheme (e.g., PSS or FDH-RSA), and elliptic-curve based scheme (e.g., EC-Schnorr).

Table 1: Comparison of security

Schemes	Security against CMA	Number-theoretical assumption	Random function assumption
TSH-ESIGN	Secure (EUF-CMA)	AERP	Truly random
PSS or FDH-RSA	Secure (EUF-CMA)	RSA	Truly random
EC-Schnorr	Secure (EUF-CMA)	EC Discrete Log.	Truly random

We compare the processing speed of TSH-ESIGN, RSA based scheme (e.g., PSS) and elliptic curve based scheme (e.g., EC-Schnorr and EC-DSA). For each scheme we estimate the amount of work needed for signature generation and verification by calculating the amount of required modular operations in the number of 1024 bit modular multiplications.

Here, we estimated the performance based on standard techniques such as the (extended) binary method for modular exponentiation and Chinese Remainder Theorem.

We assume that the modulus n for TSH-ESIGN and RSA based scheme is 1024 bits, and the modulus for elliptic curve based scheme is 160 bits. In addition, we assume public exponent $e = 2^{16} + 1$ for RSA based scheme, and $e = 2^5$ for TSH-ESIGN.

The next table shows the evaluation.

Table 2: Comparison of computation amount

<i>Schemes</i>	<i>Sig. Gen.</i> <i>(M(1024))</i>	<i>Sig. Ver.</i> <i>(M(1024))</i>
TSH-ESIGN	9	5
RSA-based scheme (e.g., PSS or FDH-RSA)	384	17
EC-based scheme (e.g., EC-Schnorr or EC-DSA)	24	28

Here, $M(b)$ ($I(b)$) denotes the amount of work for one b -bit modular multiplication (inversion). We assume $M(b_1) = M(b_2)(b_1/b_2)^2$, $I(b_1) = I(b_2)(b_1/b_2)^2$, and $I(b)/M(b) \approx 4$.

3 Security Assessment of TSH-ESIGN

This section shows our result on the security of TSH-ESIGN (See Appendix for the proof of this theorem and the concrete analysis of the reduction cost for proving the security [4]).

Definition 3.1 Let \mathcal{G} be a key-generator of the TSH-ESIGN algorithm. Approximate e -th root problem (AERP) is, for given $pk := \{n, e\} \leftarrow \text{Gen}(1^k)$ and $y \leftarrow_R \{0, 1\}^{k-1}$, to find $x \in (\mathbb{Z}/n\mathbb{Z}) \setminus p\mathbb{Z}$ such that $0 \parallel y == [x^e \bmod n]^k$.

The approximate e -th root problem (AERP) is intractable, if for any (uniform/non-uniform) probabilistic polynomial time machine Adv , for any constant c , for sufficiently large k ,

$$\Pr[Adv(k, n, e, y) \rightarrow x] < 1/k^c,$$

where $0 \parallel y == [x^e \bmod n]^k$. The probability is taken over the coin flips of \mathcal{G} and A .

The assumption that the approximate e -th root problem (AERP) is intractable is called the approximate e -th root assumption (AERP assumption).

Theorem 3.2 TSH-ESIGN is existentially unforgeable against adaptive chosen message attacks (EUF-CMA), in the random oracle model, provided that the approximate e -th root (AERP) assumption is true.

4 Limitations

As for the limitations on the formal security proof in the random oracle model, our comments are the same as those by [1].

5 Intellectual Property Statement

NTT has filed patent applications (Japan, USA, Canada, UK, France, Germany and Italy) on the techniques used in this contribution. NTT will license any resulting patent in a reasonable and non-discriminatory fashion. A letter to this effect will be provided.

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] 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).
- [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] Fujisaki, E. and Okamoto, T.: Security of Efficient Digital Signature Scheme TSH-ESIGN, manuscript (1998 November).
- [5] Okamoto, T.: A Fast Signature Scheme Based on Congruential Polynomial Operations, IEEE Trans. on Inform. Theory, IT-36, 1, pp.47-53 (1990).

Appendix

Security of Efficient Digital Signature Scheme TSH-ESIGN

Eiichiro Fujisaki Tatsuaki Okamoto

1 Introduction

In this manuscript, we will prove the security of an efficient digital signature scheme, TSH-ESIGN.

2 Definitions

Definition 2.1 Let A be a probabilistic algorithm and let $A(x_1, \dots, x_n; r)$ be the result of A on input (x_1, \dots, x_n) and coins r . We define by $y \leftarrow A(x_1, \dots, x_n)$ the experiment of picking r at random and letting y be $A(x_1, \dots, x_n; r)$. If S is a finite set, let $y \leftarrow_R S$ be the operation of picking y at random and uniformly from finite set S . ε denote the null symbol and, for list τ , $\tau \leftarrow \varepsilon$ denote the operation of letting list τ be empty. Moreover, $\|$ denotes the concatenation operator, $|\cdot|$ denotes the operator that returns the size of bit length, i.e., $|y| := \lfloor \log_2 y \rfloor + 1$, and, for n -bit string x , $[x]^k$ and $[x]_k$ denote the most and least significant k bits of x , respectively ($k \leq n$).

Definition 2.2 [Random Oracle Model] We define by Ω the set of all maps from the set $\{0, 1\}^*$ of finite strings to the set $\{0, 1\}^\infty$ of infinite strings. $H \leftarrow \Omega$ means that we choose map H from a set of an appropriate finite length (say $\{0, 1\}^a$) to a set of an appropriate finite length (say $\{0, 1\}^b$), from Ω at random and uniformly, restricting the domain to $\{0, 1\}^a$ and the range to the first b bits of output.

Definition 2.3 [Digital Signature Scheme] A digital signature scheme is defined by a triple of algorithms (Gen, Sig, Ver) such that

- Key generation algorithm Gen is a probabilistic polynomial-time algorithm which on input 1^k ($k \in \mathbb{N}$) outputs a pair (pk, sk) , called public key and secret key.
- Signing algorithm Sig is a probabilistic polynomial-time algorithm which on input $sk \leftarrow Gen(1^k)$ and a message $m \in \{0, 1\}^*$ produces a string, $s \in \{0, 1\}^*$, called a signature of m .
- Verification algorithm Ver is a probabilistic polynomial-time algorithm which on input $pk \leftarrow Gen(1^k)$ (sk is not inputted!) and a pair of a message and a signature, (m, s) returns 0 (i.e. invalid) or 1 (i.e. valid) to indicate whether or not the signature is valid. Here we insist that, for any (m, s) where $s \leftarrow Sig_{sk}(m)$, we require $Ver_{pk}(m, s) = 1$, otherwise $Ver_{pk}(m, s) = 0$ with overwhelming probability.

Definition 2.4 [Forging Algorithm F] Let $\Pi := (Gen, Sig, Ver)$ be a digital signature scheme. Let A be an algorithm which on input pk obtained by running the generator, Gen ,

has access to random hash function H and signing oracle Sig_{sk} and eventually outputs some string. We say that an adversary A is a $(t, q_h, q_s, \epsilon(k))$ -forger for $\Pi(1^k)$ if

$$Adv_F^{\Pi}(k) := \Pr[H \leftarrow \Omega; (pk, sk) \leftarrow Gen(1^k) : F^{H, Sig_{sk}}(pk) \rightarrow (m, s)] \geq \epsilon(k),$$

where $Ver_{pk}(m, s) = 1$ and m is not included in the list of queries that F asks to $Sig_{sk}(\cdot)$, and moreover F runs within at most running time t , asking at most q_h queries to $H(\cdot)$ and at most q_s queries to $Sig_{sk}(\cdot)$.

3 TSH-ESIGN: ESIGN with Trisection Size Hash

This section introduces a digital signature scheme, TSH-ESIGN.

3.1 Trisection Size Hash function h

Although it is necessary that the hash function used in our proposed signature scheme is ideal (or random oracle) to prove the security of the signature scheme, an arbitrary hash function (e.g., MD5 and SHA-1) is available in a practical use. When the signature scheme is defined over $\mathbb{Z}/n\mathbb{Z}$, where $|n| = 3k$, the underlying hash function $h(\cdot)$ is defined as $h : \{0, 1\}^* \rightarrow \{0, 1\}^{k-1}$. (I.e., the size of the image of hash h is around one third of the size of $|n|$.)

3.2 Key Generation algorithm Gen

Key generation algorithm Gen on input security parameter 1^k ($k \in \mathbb{N}$) outputs (pk, sk) where $pk = \{n, e\}$ and $sk = \{n, e, p, q\}$. The parameters, n, e, p and q , satisfy the following condition:

- p, q ($p \neq q$) are prime numbers with k -bit length, i.e., $k = |p| = |q|$.
- $n := p^2q$ with $3k$ -bit length and $e > 4$.

3.3 Signature Generation $Sig_{sk}(m)$

The signature s of message m is computed by the signature algorithm $Sig_{sk}(\cdot)$ as follows:

Step 1 Pick r at random and uniformly from $(\mathbb{Z}/pq\mathbb{Z}) \setminus p\mathbb{Z} := \{r \in \mathbb{Z}/pq\mathbb{Z} \mid \gcd(r, p) = 1\}$.

Step 2 Set $z \leftarrow (0 \| h(m) \| 0^{2k})$ and $\alpha \leftarrow (z - r^e) \bmod n$.

Step 3 Set (w_0, w_1) such that

$$w_0 = \left\lceil \frac{\alpha}{pq} \right\rceil, \tag{4}$$

$$w_1 = w_0 \cdot pq - \alpha. \tag{5}$$

If $w_1 \geq 2^{2k-1}$, then go back to **Step 1**.

Step 4 Set $t \leftarrow \frac{w_0}{er^{e-1}} \bmod p$, and $s \leftarrow (r + tpq) \bmod n$.

Step 5 Output s .

3.4 Signature Verification $Ver_{pk}(m, s)$

For a pair of message m and signature s , the verification algorithm $Ver_{pk}(m, s)$ outputs *valid* (or 1) if

$$[s^e \bmod n]^{k+1} == 0 || h(m) || 0, \quad (6)$$

otherwise *invalid* (or 0).

Here we define by $Ver(h(m), pk)$ the set of *valid* signatures, namely, given $(h(m), pk)$, signatures that satisfy Equation (6).

Remark:

In practice, the verification equation can be replaced by

$$[s^e \bmod n]^k == 0 || h(m). \quad (7)$$

The formal proof in the next section still works with minor modification of the approximate e -th root assumption.

4 Security

In this section, we examine the security of TSH-ESIGN. We first introduce a problem that is considered to be difficult to solve and then prove that breaking our scheme is as difficult as solving this problem.

We call the underlying problem the approximate e -th root problem (AERP). Roughly speaking, AERP is the problem, for given (y, e) , finding x such that $x^e \bmod n$ is approximately equivalent to y , i.e., $x^e \bmod n \approx y$.

Now let us define the approximate e -th root problem (AERP) and the breaking algorithm.

Definition 4.1 [Approximate e -th root problem (AERP)] Let Gen be a generator of the TSH-ESIGN algorithm. Approximate e -th root problem (AERP) is, for given $pk := \{n, e\} \leftarrow Gen(1^k)$ and $y \leftarrow_R \{0, 1\}^{k-1}$, to find $x \in (\mathbb{Z}/n\mathbb{Z}) \setminus p\mathbb{Z}$ such that $0 || y || 0 == [x^e \bmod n]^{k+1}$.

Definition 4.2 [AERP-Breaker B] We say that an adversary B is a $(t, \epsilon(k))$ -AERP-breaker if

$$Adv_B^{AERP}(k) := \Pr[(pk, sk) \leftarrow Gen(1^k); y \leftarrow_R \{0, 1\}^{k-1} : B(pk, y) \rightarrow x] \geq \epsilon(k),$$

where $0 || y || 0 == [x^e \bmod n]^k$ and B runs within at most running time t .

The following theorem is the main result of this manuscript. This theorem implies that breaking AERP is as difficult as forging TSH-ESIGN (existentially forging under adaptive chosen message attacks).

Theorem 4.3 If there exists a $(t(k), q_h(k), q_s(k), \epsilon(k))$ -forger F for TSH-ESIGN, then there exists a $(t'(k), \epsilon'(k))$ -AERP-Breaker B such that

$$\begin{aligned} t'(k) &= t(k) + 4(q_h + q_s) \cdot \theta(k^3), \text{ and} \\ \epsilon'(k) &= (1/q_h) \cdot \epsilon(k). \end{aligned}$$

Proof:

Let $\Pi := (Gen, Sig, Ver)$ be a triple of the TSH-ESIGN algorithm and F be an (t, q_h, q_s, ϵ) -forging algorithm for TSH-ESIGN. We will present an AERP-Breaker B including F as an oracle. The aim of B is, given y , to find an approximate root x such that $0||y||0 == [x^e \bmod n]^{k+1}$. Recall that the advantage of ESIGN-breaker B is defined by

$$Adv_B^{\text{AERP}}(k) := \Pr[(pk, sk) \leftarrow Gen(1^k); y \leftarrow_R \{0, 1\}^{k-1} : B(pk, y) \rightarrow x].$$

Likewise recall that the advantage of a forger F for TSH-ESIGN is defined by

$$Adv_F^{\text{II}}(k) := \Pr[H \leftarrow \Omega; (pk, sk) \leftarrow Gen(1^k) : F^{H, Sig_{sk}}(pk) \rightarrow (m, s), \text{ where } s \in Ver(H(m), pk)],$$

where $Ver(H(m), pk)$ is defined as above in Sec.3.4 and m is not included in the list of queries that F asks to $Sig_{sk}(\cdot)$. Since forging algorithm F will make two kinds of queries, hash oracle queries and signing oracle queries, B must answer these queries by itself. Below we describe the specification of AERP-breaker B :

AERP-Breaker: $B(pk, y)$

```

set  $\tau \leftarrow \varepsilon$  (empty) and  $x \leftarrow \varepsilon$  (null);
set  $l \leftarrow_R \{1, \dots, q_h\}$ ,  $i \leftarrow 0$  and  $j \leftarrow 0$ ;
run  $F(pk)$ ;
do while  $F$  does not ask query  $Q$  to  $H(\cdot)$  nor ask to  $Sig_{sk}(\cdot)$ 
  if  $F$  asks query  $Q$  to  $H(\cdot)$ 
     $i++$ ;  $j++$ ; (increment  $i, j$ );
    if  $j == l$ 
      set  $Q_l \leftarrow Q$ ;
      put  $(Q_l, \varepsilon, y)$  in the list  $\tau$  and answer  $F$  with  $y$ ;
    else if  $Q \notin \tau$ 
      set  $Q_i \leftarrow Q$ ;
      do while  $0||*||0 == [x_i^e \bmod n]^{k+1}$ ,
         $x_i \leftarrow_R (\mathbb{Z}/n\mathbb{Z}) \setminus p\mathbb{Z}$ ;
      set  $(0||y_i||0) \leftarrow [x_i^e \bmod n]^{k+1}$ ;
      put  $(Q_i, x_i, y_i)$  in the list  $\tau$  and answer  $F$  with  $y_i$ ;
    else ( $Q \in \tau$ )
      answer  $F$  with the corresponding  $y' \in \tau$ ;
  else if  $F$  asks query  $Q$  to  $Sig_{sk}(\cdot)$ 
     $i++$  (increment  $i$ );
    if  $Q_i = Q_l$ 
      abort  $F$  and break;
    else if  $Q \notin \tau$ 
      set  $Q_i \leftarrow Q$ ;
      do while  $0||*||0 == [x_i^e \bmod n]^{k+1}$ 
         $x_i \leftarrow_R (\mathbb{Z}/n\mathbb{Z}) \setminus p\mathbb{Z}$ ;
      set  $(0||y_i||0) \leftarrow [x_i^e \bmod n]^{k+1}$ ;
      put  $(Q_i, x_i, y_i)$  in the list  $\tau$  and answer  $F$  with  $x_i$ ;

```


Sketch of Proof:

Recall that $Ver(h(m), pk) := \{s \in (\mathbb{Z}/n\mathbb{Z}) \setminus p\mathbb{Z} \mid [s^e \bmod n]^{k+1} == 0 \mid \mid h(m) \mid \mid 0\}$. Represent $s \in Ver(h(m), pk)$ by $s = r + tpq$ where $r \in (\mathbb{Z}/pq\mathbb{Z}) \setminus p\mathbb{Z}$, and $t \in \mathbb{Z}/p\mathbb{Z}$. Then one can easily check that (r, t) is the unique representation of s .

Let $Setr(h(m)) := \{r \in (\mathbb{Z}/pq\mathbb{Z}) \setminus p\mathbb{Z} \mid 0 \leq (-\alpha \bmod pq) + pq < 2^{2k-1}\}$ where $\alpha := ((0 \mid \mid h(m) \mid \mid 0^{2k}) - r^e) \bmod n$. Then one can make a bijective map $Ver(h(m), pk) \rightarrow Setr(h(m))$ by $r := s \bmod pq$ and $t := (-\alpha \bmod pq) + pq$ where $\alpha := ((0 \mid \mid h(m) \mid \mid 0^{2k}) - r^e) \bmod n$. Therefore, $\#Ver(h(m), pk) = \#Setr(h(m))$. \spadesuit

Lemma 4.5 *For given $s_0 \in Ver(h(m), pk)$, the following equation holds:*

$$\Pr[s \leftarrow Sig_{sk}^h(m) : s == s_0] = \frac{\Pr_{s \leftarrow_R (\mathbb{Z}/n\mathbb{Z}) \setminus p\mathbb{Z}}[s == s_0 \mid s \in Ver(h(m), pk)]}{\#Ver(h(m), pk)}, \quad (9)$$

where $\Pr_{s \leftarrow_R (\mathbb{Z}/n\mathbb{Z}) \setminus p\mathbb{Z}}[s == s_0 \mid s \in Ver(h(m), pk)]$ denotes the conditional probability of Event $[s \leftarrow_R (\mathbb{Z}/n\mathbb{Z}) \setminus p\mathbb{Z} : s == s_0]$ given Event $[s \leftarrow_R (\mathbb{Z}/n\mathbb{Z}) \setminus p\mathbb{Z} : s \in Ver(h(m), pk)]$.

Sketch of Proof:

From Lemma 4.4, the proof of this lemma is straightforward. \spadesuit

From Lemma 4.5, we found that B can answer F by itself unless F asks Q_l as a signing query. Eventually, when F outputs (Q, s) , Q is considered to be Q_j for some j . In the case of $j = l$, B succeeds to break AERP because $[s^e \bmod n]^{k+1} == y$. Then the success probability $\epsilon'(k)$ is at least $(1/q_h) \cdot \epsilon(k)$. \spadesuit