Evaluation of Security Level of Cryptography: ESIGN Signature Scheme

Alfred Menezes, Minghua Qu, Doug Stinson, Yongge Wang
Certicom Research
Contact: amenezes@certicom.com

January 15, 2001

1 Summary

The ESIGN signature scheme was proposed in 1985 by Tatsuaki Okamoto. It has been proven to be existentially unforgeable against chosen-message attacks assuming that the approximate e-th root (AER) problem is hard and that the employed hash function is a random function. While the AER problem has been studied by some researchers, it has not received as much attention as the integer factorization problem or the discrete logarithm problem. One way to solve the AER problem is to factor the integer n, where $n = p^2q$ and p and q are primes of the same bitlength. The parameters recommended ensure that ESIGN resists all known attacks for factoring integers of this form.

2 Protocol specification

2.1 ESIGN key pairs

For the security parameter $k = pLen$, each entity does the following:

1. Randomly select two distinct primes, p, q, each of bitsize k and compute $n = p^2q$.
2. Select an integer $e > 4$.
3. A's public key is (n, e, k); A's private key is (p, q).

In addition, one needs to specify a hash function H' whose output length is k bits.

2.2 ESIGN signature generation

To sign a message m, an entity A with the private key (p, q) does the following:

1. Compute $H'(m)$, and let $H(m)$ be obtained from $H'(m)$ by deleting the most significant bit.
2. Pick r uniformly at random from $\{r \in Z_{pq} : \gcd(r, p) = 1\}$.
3. Set $z = (0 \| H(m) \| 0^{2k})$ and $\alpha = (I(z) - r^e) \bmod n$, where the function $I()$ converts a binary string into an integer in the usual way.
4. Set (w_0, w_1) such that

$$w_0 = \left\lceil \frac{\alpha}{pq} \right\rceil, \quad (1)$$

$$w_1 = w_0 pq - \alpha. \quad (2)$$

5. If $w_1 \geq 2^{2k-1}$, then go back to step 2.
6. Set $t = \frac{w_0}{er^{e-1}} \bmod p$ and $s = B_{3k}[(r + tpq) \bmod n]$, where $B_{3k}[X]$ converts an integer X to a binary string of length $3k$ (by prepending 0's if necessary).
7. Output the signature s.

2.3 ESIGN signature verification

To verify A's signature s on m, B obtains an authentic copy of A's public key (n, e, k). B then does the following:

1. Compute $H'(m)$, and let $H(m)$ be obtained from $H'(m)$ by deleting the most significant bit.
2. Check whether the following equation holds:

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

   where the function $[X]^k$ takes the most significant k bits of the input X.
3. If it holds, output "valid"; otherwise output "invalid".

Note: The signature generation algorithm computes the integer s such that $s^e \bmod n$ lies in a certain interval determined by the message. The signature verification algorithm demonstrates that $s^e \bmod n$ does indeed lie in the specified interval.

Note: In order to speed up the signing time, step 5 in the signature generation process could be optional. The condition $w_1 \geq 2^{2k-1}$ ensures the value $w_1 = s^e - I(z)$ is uniformly distributed in the interval $[0, 2^{2k+1} - 1]$ as required by the security proof for ESIGN. However, no security weakness is known if the test is omitted.

Performance note: Fujioka, Okamoto, and Miyaguchi [7] describe an implementation of ESIGN (with $e = 32$) which suggests that it is twenty times faster than RSA signatures (with $e = 2^{16} + 1$) with comparable key and signature lengths.

3 Security level of cryptographic techniques

The security objective of any signature scheme is to be existentially unforgeable against a chosen-message attack. The goal of an adversary who launches such an attack against a legitimate entity A is to obtain a valid signature on a single message m, after having obtained A's signature on a collection of messages (not including m) of the adversary's choice.

ESIGN bases its security on the AER (approximate e-th root) assumption which is defined next. Let $n = p^2q$, where p and q are primes of the same bitlength. The *AER problem* is to find x, given n, e, y, such that a portion (specifically, the $|n|/3$-most significant bits) of $x^e \bmod n$ equals the corresponding part of y, i.e., $[x^e \bmod n]^{|n|/3} = [y]^{|n|/3}$. The *AER assumption* is that there is no efficient algorithm for solving the AER problem.

The submission includes a proof that the ESIGN signature scheme is existentially unforgeable against adaptive chosen-message attacks in the random oracle model under the AER assumption. In the random oracle model, the hash function employed is modeled by a random function.

The security proof of the ESIGN signature scheme is similar to the security proof of Bellare and Rogaway for the Full Domain Hash (FDH) RSA signature scheme. The proof guarantees resistance to attacks that do not use specific properties of the hash function, assuming only that the AER problem is intractable. As with all proofs in the random oracle model, it should not be regarded as an unconditional security proof due to known (albeit theoretical) pitfalls in the random oracle model (see [5]).

4 Security level of cryptographic primitive functions

4.1 Attacks on the hash function

DEFINITION. A *(cryptographic) hash function H* is a function that maps bit strings of arbitrary lengths to bit strings of a fixed length t such that:

1. H can be computed efficiently;
2. (*preimage resistance*) For y selected uniformly at random from $\{0,1\}^t$, it is computationally infeasible to find a bit string x such that $H(x) = y$.
3. (*second preimage resistance*) Given x_1, it is computationally infeasible to find a different bit string x_2 such that $H(x_1) = H(x_2)$.
4. (*collision resistance*) It is computationally infeasible to find distinct bit strings x_1 and x_2 such that $H(x_1) = H(x_2)$.

HASH FUNCTION SECURITY REQUIREMENTS. The following explains how attacks on ESIGN can be successfully launched if hash function is not collision resistant or second preimage resistant.

1. If the hash function is not collision resistant, then an entity A may be able to repudiate signatures as follows. A first generate two messages m and m' such that $H(m) = H(m')$; such a pair of messages is called a *collision* for H. She then signs m, and later claims to have signed m' (note that every signature for m is also a signature for m').

2. If H is not collision resistant, then an entity B could use the birthday attack to swindle A as follows. B prepares two versions (m_1 and m_2) of a contract, where m_1 is favorable to A and m_2 bankrupts A. B makes several subtle changes to each document and compares whether $H(m_1') = H(m_2')$, where m_1' (m_2') is a subtle variant of m_1 (m_2). When B finds two such variants, B gets the signature s of m_1' from A. Then s is also a signature of m_2'.

3. If the hash function H is not second preimage resistant, then an entity B may be able to forge signatures as follows. B generates a message m, find another message m' such that $H(m) = H(m')$, and gets the signature s of m' from A. Then s is also a signature of m.

IDEAL SECURITY. A t-bit hash function is said to be have *ideal security* [9] if both: (i) given a hash output, producing a preimage (or a second preimage) requires approximately 2^t operations; and (ii) producing a collision requires approximately $2^{t/2}$ operations (this is the best that can be hoped for, in view of the birthday attacks). The hash function used in ESIGN is a k-bit hash function and is required to have ideal security. The fastest method known for attacking ESIGN by exploiting properties of hash function is to find collisions for the hash function. Since this is believed to take $2^{k/2}$ steps, attacking ESIGN in this way is computationally infeasible. Note, however, that this attack imposes an upper bound of $2^{k/2}$ on the security level of ESIGN, regardless of the size of the primary security parameter n. Of course, this is also the case with all present signature schemes with appendix.

5 Security level of cryptographic primitive problem: the approximate eth roots problem

ESIGN was originally proposed by Okamoto and Shiraishi [11] and was motivated by the digital signature mechanism OSS devised by Ong, Schnorr, and Shamir [12]. The OSS scheme was shown to be insecure by Pollard in a private communication to them. Ong, Schnorr, and Shamir [12] modified their original scheme but this too was shown insecure by Estes et al. [6].

ESIGN bases its security on the AER (approximate e-th root) assumption which is defined next. Let $n = p^2q$, where p and q are primes of the same bitlength. The *AER problem* is to find x, given n, e, y, such that a portion (specifically, the $|n|/3$-most significant bits) of $x^e \bmod n$ equals the corresponding part of y, i.e., $[x^e \bmod n]^{|n|/3} = [y]^{|n|/3}$. The *AER assumption* is that there is no efficient algorithm for solving the AER problem.

The original version of ESIGN [11] proposed $e = 2$ as the appropriate value for the public key. Brickell and DeLaurentis [3] demonstrated that this choice was insecure. Their attack also extends to the case $e = 3$; see Brickell and Odlyzko [4]. Okamoto [10] then revised the method by requiring $e \geq 4$. No weakness for these values of e have been reported in the literature.

The problem is related to the *RSA problem* which is one of finding x, given n, e, y, such that $y \equiv x^e \pmod{n}$. Clearly, if one can efficiently solve the RSA problem then one can also

efficient solve the AER problem. The submission conjectures that the RSA and AER problems are polynomial-time equivalent (and that the integer factorization and AER problems and polynomial-time equivalent when e is even); however no proof of this is known.

One way in which an adversary can break the scheme is to compute A's private key (p,q) from A's public key (n,e,k). If this is done, the adversary can subsequently forge A's signature on any message of its choice.

Problem Definition. The *integer factorization problem* is the following: given an integer n, determine the factorization of n.

Known algorithms for integer factorization

This subsection overviews the algorithms known for factoring and discusses how they can be avoided in practice.

1. **Trial division.** Once it is established that an integer is composite, before expending vast amount of time with more powerful techniques, the first thing that should be attempted is trial division by all "small" primes. Here "small" is determined as a function of the size of n. As an extreme case, trial division can be attempted by all primes up to $\sqrt{n}$. If this is done, trial division will completely factor n but the procedure will take roughly $\sqrt{n}$ divisions in the worst case when n is a product of two primes of the same size.

2. **Pollard's rho factoring algorithm.** Pollard's rho algorithm is a special-purpose factoring algorithm for finding small factors of a composite integer. Let $f : S \to S$ be a random function, where S is a finite set of cardinality m. Let x_0 be a random element of S, and consider the sequence $x_0, x_1, \ldots$ defined by $x_{i+1} = f(x_i)$ for $i \geq 0$. Since S is finite, the sequence must eventually cycle, and consists of a tail of expected length $\sqrt{n\pi/8}$ followed by an endless repeating cycle of expected length $\sqrt{n\pi/8}$. A problem that arises in the factorization problem is of finding distinct indices i and j such that $x_i = x_j$. The expected time for finding such a collision is $O(\sqrt{p})$ where p is a prime factor of n.

3. **Pollard's p-1 factoring algorithm.** Pollard's $p-1$ factoring algorithm is a special-purpose factoring algorithm that can be used to efficiently find any prime factors p of a composite integer n for which $p-1$ is smooth with respect to some relatively small B. Let n be an integer having a prime factor p such that $p-1$ is B-smooth. The running time of Pollard's $p-1$ algorithm for finding the factor p is $O(B \ln n / \ln B)$ modular multiplication.

4. **Elliptic curve factoring.** The elliptic curve factoring method (ECM) has an expected running time of $O(exp((\sqrt{2}+o(1))(\ln p)^{1/2}(\ln\ln p)^{1/2}))$ to find a prime factor p of n. It is the fastest factoring algorithm known when the running time is measured in terms of

the smallest prime factor of n, and is thus useful for finding factors of n in some situations when n has a prime factor that is significantly smaller than the other prime factors of n.

5. **Quadratic sieve factoring.** The quadratic sieve factoring algorithm has the same expected running time as the elliptic curve factoring algorithm in the special case when n is the product of two primes of equal size. However, for such numbers, the quadratic sieve is superior in practice.

6. **Number field sieve factoring.** The general version of the number field sieve factoring algorithm has an expected running time of $O(exp((\sqrt{c}+o(1))(\ln n)^{1/3}(\ln\ln n)^{2/3}))$ to factor n, where c is approximately 1.923. This is asymptotically superior to the quadratic sieve factoring method. This superiority has also been validated by numerous researchers through extensive experimentation.

7. **Factoring $n = p^2q$.** Although it is not known whether $n = p^2q$ is easier to factor than $n = pq$, some special algorithms to factor $n = p^2q$ have been studied in [1, 13]. These techniques are specific to the elliptic curve factoring method, and are just several times faster than the traditional ECM. The fastest algorithm known for factoring both integers n of the form $n = p^2q$ and $n = pq$ (where the primes p and q have the same bit length) is the number field sieve (NFS) method, whose running time depends only on the bit length of n.

8. **Factoring $n = p^rq$.** Boneh, et al. [2] presented an algorithm for factoring $n = p^rq$ with large r using the LLL algorithm. Their algorithm, however, is only effective for the case where r is large (at least $(\log p)^{1/2}$). If r is constant or small, the running time of their algorithm is exponential in the bitsize of n. For $n = p^2q$, their algorithm is less efficient than the ECM and NFS methods.

Summary: If n is carefully chosen, namely $n = p^2q$ where p and q are randomly chosen primes of the same bitlength which is at least 320 (so n has bit length at least 960), then all known algorithms for factoring n are thwarted.

6 Recommended parameters

The recommended ESIGN parameters are as follows:

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

In Table 1, the security levels of ESIGN with these parameters are compared with other submitted schemes, with RSA, and with symmetric schemes (e.g., the Advanced Encryption Standard–AES). Some of the comparisons among symmetric key cryptography, RSA security, and ECDSA security are adapted from Lenstra and Verheul [8].

Table 1: Rough Comparison of Security Levels

Symmetric Key Size	RSA Modulus Size	ESIGN Modulus Size	ACE Modulus Size	ECDSA Key Size	MY-ELLTY Key Size
40					160
48					192
76	960	960	960	152	
80 (SKIPJACK)	1024	1024	1024	160	
112 (Triple-DES)	2048	2048	2048	224	
128 (128-bit AES)	3072	3072	3072	256	
192 (192-bit AES)	7680	7680	7680	384	
256 (256-bit AES)	15360	15360	15360	512	

7 Performance comparison

Table 1 presented comparable key lengths of several schemes. Generally, the ESIGN signature scheme is faster than both ECDSA and RSA signature schemes with comparable key lengths. ACE, RSA, and ESIGN have roughly the same public key size for comparable security level. ACE and RSA have roughly the same private key size for comparable security level. ESIGN's private key size is roughly $2/3$ of the ACE (or RSA) private key size for comparable security level. ECDSA and MY-ELLTY have significantly smaller key size for comparable security level. However, the hash function used in MY-ELLTY is not collision resistant, thus it is not secure against attacks on the hash function.

References

[1] L.M. Adleman and K.S.McCurley. Open problems in number theoretic complexity II, *Proc. of ANTS I*, LNCS 877, pages 291–322, 1995.

[2] D. Boneh. G. Durfee, and N. Howgrave-Graham. Factoring $N = p^r q$ for large r. *Advances in Cryptology, Crypto 99*, LNCS 1666, pages 326–337, 1999.

[3] E.F. Brickell and J.M. DeLaurentis. An attack on a signature scheme proposed by Okamato and Shiraishi. *Advances in Cryptology, Crypto 85*, LNCS 218, pages 28–32, 1986.

[4] E.F. Brickell and A.M. Odlyzko. Cryptanalysis: A survey of recent results. G.J. Simmons, editor, *Contemporary Cryptology: The Science of Information Integrity*, pages 501–540, IEEE Press, 1992.

[5] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. *Proc. of the STOC*, pages 209–218, ACM Press, 1998.

[6] D. Estes, L.M. Adleman, K. Kompella, K.S. McCurley, and G.L. Miller. Breaking the Ong-Schnorr-Shamir signature scheme for quadratic number fields. *Advances in Cryptology, Crypto 85*, LNCS 218, pages 3–13, 1086.

[7] A. Fujioka, T.Okamoto, and S. Miyaguchi. ESIGN: An efficient digital signature implementation for smart cards. *Advances in Cryptology, Eurocrypt 91*, LNCS 547, pages 446–457, 1991.

[8] A. Lenstra and E. Verheul. Selecting cryptographic key sizes. Distributed in the *3rd workshop on Elliptic Curve Cryptography (ECC 99)*. Available from http://www.cryptosavvy.com/

[9] A. Menezes, P. van Oorschot and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.

[10] T. Okamoto. A fast signature scheme based on congruential polynomial operations. *IEEE Transactions on Information Theory*, **36**:47–53, 1990.

[11] T. Okamoto and A. Shiraishi. A fast signature scheme based on quadratic inequalities. *Proceedings of the 1985 IEEE Symposium on Security and Privacy*, pages 123–132, 1985.

[12] H. Ong, C.P. Schnorr, and A. Shamir. An efficient signature scheme based on quadratic equations. *Proceedings of the 16th Annual ACM Symposium on Theory pf Computing*, pages 208–216, 1984.

[13] R. Peralta and E. Okamoto. Faster factoring of integers of a special form, *IEICE Trans. Fundamentals*, E79-A, **4**:489–493, 1996.