Evaluation of Security Level of Cryptography: ACE Signature Scheme

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

January 15, 2001

1 Summary

The ACE signature scheme is an RSA-based signature scheme that has some attractive proofs of security. It has been proven existentially unforgeable against chosen-message attacks assuming that the strong RSA problem is hard and that the UOWHF is universally collision-resistant. It has also been proven existentially unforgeable against chosen-message attacks assuming that the RSA problem is hard and that the UOWHF is a random function. While the RSA and strong RSA problems are known to be no harder than the integer factorization problem, it is not known whether the RSA and strong RSA problems are as hard as the integer factorization problem. In fact, the only evidence to date suggests that the RSA problem may in fact be easier than the integer factorization problem.

2 Protocol specification

2.1 ACE key pairs

The signature scheme defined in the submission employs two key types, whose representation consists of the following tuples:

ACE Signature public key: (N, h, x, e', k', s).

ACE Signature private key: (p, q, a).

For a given parameter m, with $1024 \leq m \leq 16384$, the components are as follows.

- p: $\lfloor m/2 \rfloor$-bit prime with $(p-1)/2$ also a prime.
- q: $\lfloor m/2 \rfloor$-bit prime with $(q-1)/2$ also a prime.
- $N = pq$, and has either m or $m-1$ bits.
- h, x are quadratic residue modulo N.
- e' is a 161-bit prime number.
- a is an element of $\{0, \ldots, (p-1)(q-1)/4-1\}$.
- k' is an element of $\mathbf{B}^{184}$, that is, k' is a concatenation of 184 elements from $\mathbf{B}$, where $\mathbf{B}$ is the set of all 256 bytes.
- s is an element of $\mathbf{B}^{32}$.

In the following, we will use the following notations:

- $L(M)$ denotes the length of the binary string M.
- $L_b(m)$ denotes the length of the binary representation of m, where m is an integer.
- $L_B(m) = 8\lceil m/8 \rceil$, where m is an integer.

2.2 ACE signature generation

A signature of the ACE signature scheme has the form $(d, w, y, y', \tilde{k})$ and is encoded as a byte-string. (For a description of the encoding method, please see the submission.)

Input: A public key (N, h, x, e', k', s), the corresponding private key (p, q, a), and a byte string $M \in \mathbf{B}^*$, $0 \leq L(M) \leq 2^{64}$.

Output: A byte string encoded signature $\sigma \in \mathbf{B}^*$ of M.

1. Perform the following steps to hash the input data:

   1.1. Generate a hash key $\tilde{k} \in \mathbf{B}^{20m+64}$ at random, such that
   $$m = L_b(\lceil (L(M)+8)/64 \rceil).$$

   1.2. Compute $m_h \leftarrow I^{Z}_{\mathbf{W}^*}(\text{UOWHash}''(\tilde{k}, M))$, where

   1.2.1. $\mathbf{W}$ is the set of words.

   1.2.2. $I^{Z}_{\mathbf{W}^*}$ is a conversion operator from $\mathbf{W}^*$ to integers.

   1.2.3. UOWHash$''$ is a universal one-way hash function.

2. Select $\tilde{y} \in \{1, \ldots, N-1\}$ at random, and compute $y' \leftarrow \tilde{y}^2 \bmod N$.
3. Compute $x' \leftarrow (y')^{e'} h^{m_h} \bmod N$.
4. Generate a random prime e, $2^{160} < e < 2^{161}$, and its certificate of correctness (w, d) (for details see the submission).
5. Set $r \leftarrow \text{UOWHash}'''(k', L_B(N), x', \tilde{k}) \in Z$, note that $0 \leq r \leq 2^{160}$, where UOWHash$'''$ is a special-purpose hash function.
6. Compute $y \leftarrow h^b \bmod N$, where
   $$b \leftarrow e^{-1}(a - r) \bmod p'q',$$
   and $p' = (p-1)/2$, $q' = (q-1)/2$.
7. Encode the signature $(N, d, w, y, y', \tilde{k})$ as a byte-string σ.
8. Output the signature σ.

2.3 Signature verification

Input: A public key: (N,h,x,e',k',s), a signature $\sigma \in \mathbf{B}^*$, and a message $M \in \mathbf{B}^*$.

Output: If σ is a valid signature on M under the given public key, then outputs “Accept”; otherwise outputs “Reject”.

1. Decode the signature σ into tuple: $(d,w,y,y',\tilde{k})$
   - 1.1. If $L(M) \geq 2^{64}$ then stop processing and signal “Reject”.
   - 1.2. If $L(\sigma) < 85 + 2L_B(N)$ then stop processing and signal “Reject”.
   - 1.3. Compute the tuple $(d,w,y,y',\tilde{k})$.
2. Set $e \leftarrow \mathit{VerCertPrime}(s,d,w)$ (see the submission).
3. If $e =$ “Reject”, return “Reject”.
4. If $e = e'$, return “Reject”.
5. If $y = 0$ or $y \geq N$ or $y' = 0$ or $y' \geq N$ then return “Reject”.
6. Perform the following steps to hash the input data:
   - 6.1. If $L(\tilde{k}) \neq 20m + 64$, where $m = L_b(\lceil (L(M)+8)/64 \rceil)$, then return “Reject”.
   - 6.2. Compute $m_h \leftarrow I^{Z}_{W^*}(\text{UOWHash}''(\tilde{k},M))$.
7. Compute $x' \leftarrow (y')^{e'} h^{m_h} \bmod N$.
8. Set $r \leftarrow \text{UOWHash}'''(k',L_B(N),x',\tilde{k}) \in Z$, note that $0 \leq r \leq 2^{160}$.
9. If $x \neq y^e h^r (\bmod N)$ then return “Reject”.
10. Return “Accept”.

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.

The security of the ACE signature scheme is based on the strong RSA assumption. The *strong RSA problem* is as follows: Given a randomly generated RSA modulus N, and a randomly generated integer z, find y and $r > 1$ such that $y^r \equiv z \pmod{N}$. The *strong RSA assumption* says that there is no efficient algorithm for solving the strong RSA problem.

Chosen-message attacks. It is proven in [4] that the ACE signature scheme is secure against adaptive chosen-message attacks if the strong RSA assumption holds, and if UOWHF is universally collision-resistant (see §4.1).

Random oracle and chosen-message attacks. The *RSA problem* is the following: Given a randomly generated RSA modulus N, an exponent r, and a random z, find y such that $y^r \equiv z \pmod{N}$. The *RSA assumption* says that there is no efficient algorithm for solving the RSA problem. It is shown in [4] that the ACE signature scheme is secure against adaptive chosen-message attacks in the random oracle model if the RSA assumption holds. The proof guarantees resistance to attacks that do not use specific properties of the hash function, assuming only that the RSA problem is intractable.

Comparison of assumptions. On the one hand, the first proof leads to stronger assurances than the second proof because collision resistance of the UOWHF is a much weaker assumption that the random oracle assumption. On the other hand, the second proof leads to stronger assurances than the first proof because the RSA assumption is a weaker assumption than the strong RSA assumption. The difficulty of the RSA and strong RSA problems is considered further in §5.

4 Security level of cryptographic primitive functions

4.1 Attacks on the universal one-way hash function

DEFINITION. A *universal one-way hash function* is a keyed hash function that maps bit strings of arbitrary lengths to bit strings of a fixed length t such that:

1. For every K, H_K can be computed efficiently;
2. (*universal collision resistance*) If an adversary chooses a message x, and then a key K is chosen at random and given to the adversary, it is hard for the adversary to find a different message $x' \neq x$ such that $H_K(x) = H_K(x')$.

HASH FUNCTION SECURITY REQUIREMENTS. The following explains how attacks on ACE can be successfully launched if the universal hash function is not secure.

1. If for some K the hash function H_K is not collision resistant, then an entity A may be able to repudiate signatures as follows. A first generates two messages m and m' such that $H_K(m) = H_K(m')$; such a pair of messages is called a *collision* for H_K. 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 the hash function H is not universal collision resistant, then an entity B may be able to forge signatures as follows. B generates a message m, obtains the signature σ of m from A. Assume that A has signed m with H_K, then B finds another message m' such that $H_K(m) = H_K(m')$. Now σ is also a signature of m'.

IDEAL SECURITY. A t-bit universal hash function is said to be have *ideal security* if both: (i) given a hash output $H_K(x)$, producing a preimage requires approximately 2^t operations; and (ii) for any given K, producing two messages m and m' such that $H_K(m) = H_K(m')$ requires approximately $2^{t/2}$ operations. The universal hash function in ACE is a 160-bit hash function and is required to have ideal security. The fastest method known for existentially forgery attacks on ACE by exploiting properties of universal hash functions is to find a universal collision. Since this is believed to take 2^{160} steps, attacking ACE in this way is computationally infeasible. The fastest method known for repudiation attacks on ACE by exploiting properties of a universal hash function is to find a K, and then find two messages m and m' with $H_K(m) = H_K(m')$. Since this is believed to take 2^{80} steps, attacking ACE in this way is computationally infeasible. Note, however, that this attack imposes an upper bound of 2^{160} (or 2^{80} for repudiation attacks) on the security level of ACE, regardless of the size of the primary security parameter n.

5 Security level of cryptographic primitive problems: the RSA and strong RSA problems

The strong RSA assumption was first introduced in [1], and has subsequently been used in the analysis of several cryptographic schemes (see, e.g., [5, 6]). This is a potentially stronger assumption than the RSA assumption, but not much work has been published on either problem. At present, the only known method for solving either the RSA problem or the strong RSA problem is to solve the integer factorization problem.

The only result ever proved on the relationship between the RSA problem and the integer factorization problem is a negative one: Boneh and Venkatesan [3] proved that any polynomial-time reduction algorithm from the integer factorization problem to the RSA problem (with small encryption exponent e) that uses only algebraic operations can be efficiently converted to a polynomial-time algorithm for integer factorization (which doesn't make use of the hypothetical oracle for the RSA problem). This provides some evidence that the RSA problem (and consequently also the strong RSA problem) may in fact be *easier* than the integer factorization problem.

Now, if an adversary can succeed in computing A's private key (p,q) from A's public key (n,e,k), then the adversary can subsequently forge A's signature on any message of its choice. The following summarizes the state-of-the-art in our knowledge for integer factorization.

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

Known algorithm 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 $f : S \rightarrow S$ 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 (NFS) 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 $n = p^r q$.** Boneh, et al. [2] presented an algorithm for factoring $n = p^r q$ 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 $n = p^2 q$, their algorithm is less efficient than the ECM and NFS methods.

Summary: For carefully chosen n, the integer factorization problem is widely believed to be intractable. In particular, the integer n specified in the ACE signature scheme avoid all known existing factoring attacks.

6 Recommended parameters

In Table 1, the security levels of the ACE 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 [7].

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 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] N. Baric and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In: *Advances in Cryptology, Eurorypt 97*, pages 480–494, 1997.

[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] D. Boneh and R. Venkatesan, Breaking RSA may not be equivalent to factoring, *Advances in Cryptology, Eurocrypt 98*, LNCS 1403, pages 59–71, 1998.

Advances in Cryptology, Crypto 99, LNCS 1666, pages 326–337, 1999.

[4] R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. In: *6th ACM Conf. on Computer and Communication Security*, 1999.

[5] E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In: *Advances in Cryptology, Crypto 99*, 1999.

[6] R. Gennaro, S. Halevi, and T. Rabin. Secure hash-and-sign signatures without the random oracle. In: *Advances in Cryptology, Crypto 99*, pages 123–139, 1999.

[7] 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/

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