ESIGN: An Efficient Digital Signature Implementation for Smart Cards

Atsushi Fujioka Tatsuaki Okamoto Shoji Miyaguchi

NTT Laboratories
Nippon Telegraph and Telephone Corporation
1-2356, Take, Yokosuka-shi, Kanagawa-ken, 238-03 Japan

Abstract

ESIGN is an efficient digital signature algorithm [OkS, Ok], whose computation speed is more than twenty times faster than that of the RSA scheme, while its key length and signature length are comparable to those of the RSA scheme. This paper presents a software implementation of ESIGN on an 8bit micro-processor smart card. This realizes a computation time for signature generation of about 0.2 seconds. To achieve this remarkable speed for signature generation, appropriate implementation techniques such as pre-computation and table look-up techniques are effectively used. Moreover, this software implementation is compact enough for smart cards; the program size and the data size including the work area are at most 3Kbytes each. Practical identification schemes based on ESIGN are also presented.

1 Introduction

Recently, smart cards are the basis of many services such as prepaid cards, banking cards, and credit cards. In these applications, the integrity and authenticity of digital messages and documents are often required to be ensured by digital information in place of manual signatures written on paper documents. These functions can usually be achieved through the use of *digital signatures*, and they should be implemented through software.

D.W. Davies (Ed.): Advances in Cryptology - EUROCRYPT '91, LNCS 547, pp. 446-457, 1991.

Many digital signature schemes have been developed since Diffie and Hellman's seminal paper on public key cryptosystems [DH] was presented in 1976. Among these schemes, the RSA scheme [RSA] and the Fiat-Shamir signature scheme [FS] are the most respected schemes from the practical viewpoint. However, the RSA scheme has the disadvantage of very low processing speed. Usually a specific hardware chip is needed to perform the RSA function [Br]. If the RSA scheme is implemented in software on an 8bit microprocessor smart card, one complete cycle takes at least several minutes. On the other hand, although the Fiat-Shamir signature scheme is more efficient than the RSA scheme, signature generation still takes more than several seconds with a software implementation on an 8bit microprocessor smart card, even if the fast implementation technique introduced by [OnS] is used*. Moreover, the Fiat-Shamir scheme involves a trade-off between signature size and key size, and their product is relatively large. For example, when the key size equals that of the RSA scheme, then the signature size is at least seventy times as long as that of the RSA scheme†. Although a modified scheme which overcomes this trade-off problem has been proposed [GQ, OhO], it is somewhat slower than the original scheme.

In 1985, Okamoto *et al.* proposed a fast signature scheme based on polynomial operations [OkS]. The new scheme had significantly better efficiency. It is much (e.g., more than twenty times even with unsophisticated implementation) faster than the RSA scheme, and its key and signature lengths are comparable to those of the RSA scheme. Many versions of this signature scheme were suggested in [Ok], and, hereafter, we will simply call the most typical of them ESIGN (**Section 2**).

In this paper, we report the remarkable performance of our software implementation of ESIGN on an 8bit microprocessor smart card. In our implementation, a pre-computation technique plays an essential role in increasing the speed of signature generation. In addition, residue and division operations by the table look-up method utilizing the EEPROM area is also a key technique in our implementation. By utilizing both techniques, the computation time for signature generation is just about 0.2 seconds (effectively instantaneous for humans)‡. Thus we can ignore the time taken for signature generation in almost all smart card applications.

As for identification schemes, the Fiat-Shamir identification [FS] scheme is currently

*It seems to take more than 20 seconds in the implementation of [Kno], because roughly the Fiat-Shamir *signature* scheme is more than three times slower than the Fiat-Shamir *identification* scheme, which is reported to take about 6 seconds.

†Note that the trade-off level of the Fiat-Shamir *identification* scheme is often less than that of the signature scheme, because identification is done under real-time circumstances.

‡Note that in this estimation we select parameters offering a security level equivalent to that of the RSA scheme with a 64byte (512bit) composite modulus.

the most promising. In this paper, we also introduce practical identification schemes based on ESIGN, which are at least as efficient as the Fiat-Shamir identification scheme.

2 ESIGN

2.1 Notations

Z_n denotes the set of numbers between 0 and $n-1$, and Z_n^* denotes the set of numbers between 0 and $n-1$ which are relatively prime to n. $\lceil M \rceil$ denotes the least integer which is larger than or equal to M. $\lfloor M \rfloor$ denotes the greatest integer which is less than or equal to M. $x \equiv y \pmod{n}$ denotes that n divides $x-y$. $f(x) \bmod n$ denotes an integer such that n divides $f(x) - (f(x) \bmod n)$ and $f(x) \bmod n \in Z_n$. $x/y \bmod n$ denotes an integer such that n divides $x - y(x/y \bmod n)$ and $x/y \bmod n \in Z_n$. $|X|_b$ denotes $\lceil \log_b X \rceil$. In other words, $|X|_2$ denotes the bit size of X, and $|X|_{64}$ denotes the byte size of X. $a||b$ denotes the concatenation of a and b.

2.2 Procedures

- *Keys, message and signature:*
  - *Secret key:* large prime numbers P, Q $(P > Q)$.
  - *Public key:* positive integers $N = P^2Q$.
  - *Message:* a positive integer M.
  - *Signature:* an integer $S \in Z_N^*$.
- *Key generation:*
  - When originator A joins the system, he generates the secret and public keys and publishes the public key. The secret key is known only to A.
- *Signature generation:*
  - A signature S of a message M is computed by originator A as follows:
    * Pick a random number $X \in Z_{PQ}^*$.
    * Compute S such that

$$W = \left\lceil \frac{H(M) - (X^K \bmod N)}{PQ} \right\rceil,$$

$$Y = W/(KX^{K-1}) \bmod P,$$

$$S = X + YPQ,$$

where H is a one-way hash function ($H(M) \in Z_N$ for any positive integer M), K is an integer ($K \geq 4$). Function H and parameter K can be fixed in the system.

- *Signature sending:*
  - o After generating the signature S, originator A sends message M along with S to receiver B, who verifies it.
- *Signature verification:*
  - o The signature message (S, M) is considered valid if the following verification inequality holds.

$$H(M) \leq S^K \bmod N < H(M) + 2^{2\cdot|N|_2/3}.$$

2.3 Security of ESIGN

The quadratic version (parameter K is 2) was broken by Brickell and DeLaurentis in 1985 ([Ok], page 49), several months after this signature scheme was presented [OkS]. However, no attack on the higher degree versions (K is more than 3) has been reported in the last 5 years, although many excellent researchers such as Brickell, Girault, Shamir, and Vallée attempted this ([Ok], pp.49-50). We conjecture that to break our higher degree version (ESIGN) is as hard as factoring N ([Ok] p.52).

3 Implementation on Smart Cards

3.1 Preparation

To implement ESIGN on a smart card, we must determine the parameters carefully because a smart card doesn't have high processing speed or much memory.

In our implementation, K is chosen as a power of 2, i.e., 4, 8, 16, and so on. Our implementation target consists of the following steps:

- $X \in Z^*_{PQ}$,
- $F = X^K \bmod N$,
- $W = \left\lceil \dfrac{H(M) - F}{PQ} \right\rceil$,

- $G = KX^{K-1} \bmod P$,
- $Y = W/G \bmod P$,
- $S = X + YPQ$.

We choose 16, 24, 32, and 40bytes as the lengths of prime P in order to appraise the performance of ESIGN on a smart card.

Our software implementation works on a smart card with 8bit microprocessor that has clock cycle of 5MHz. The memory area consists of 384bytes RAM, 10Kbytes ROM and 8Kbytes EEPROM.

Our implementation is not straightforward. The residue and division operations take more time than addition or multiplication. Therefore, we developed an iterated table look-up method to reduce the processing time of the residue and division operations. In addition, the signature generation procedure of ESIGN has the good property that it contains some computations that do not depend on the message to be signed. Hence, when a smart card is idling, it can do pre-computation and store the computation value, to generate a signature later. This is based on the same idea as "on-line/off-line digital signatures" [EGM].

In the following subsections, we will discuss the table look-up method and pre-computation. Finally proposed implementation is appraised.

3.2 Hash Functions

We can use any secure hash function H for implementing ESIGN [Da]. In our implementation, we adopt the hash function named N-Hash [MOI] because of the compactness of its program and high execution speed. That is, the program size is about 300bytes (to be stored in ROM), and the execution speed is about 150Kbps (bits per second) in our smart card implementation, which implies that $H(M)$ can be computed within 0.06 seconds when the size of M is about 1Kbytes.

3.3 Iterated Table Look-up Method

Residue and division operations take much more processing time than addition or multiplication. To increase the computation speed of residue and division operations, several table look-up methods have been presented [KH]. However, these methods need a large amount of memory for storing the table, therefore, they are not suitable for typical smart card implementations. This subsection introduces a new table look-up method for these operations, in which all primitive operations are performed on

8bit words. It is suitable for smart card implementation, because our method needs only 9 values in the table and is based on 8bit word primitive operations for 8bit microprocessors of smart cards.

In ESIGN, residue and division operation are performed in the following steps:

- $F = X^K \bmod N$,
- $W = \left\lfloor \dfrac{H(M) - F}{PQ} \right\rfloor$,
- $G = KX^{K-1} \bmod P$,
- $T = 1/G \bmod P$.

In these steps, modulus P and N are fixed numbers for the smart card, and the divider PQ is also a fixed number. The residue table is stored in EEPROM.

Our residue table regarding modulus N consists of 9 one-bit-shifted numbers of a modulus, $N_1, \ldots, N_9$, such that $N_i = N \cdot 2^{z(N)+9-i}$, $|N_i|_2 = |N|_2 + 8$, $(i = 1, \ldots, 9)$, where $z(N)$ means the number of zeros succeeding the most significant bit of N.

For example, if the modulus number N is 11011010 00001001 (DA09 in hexadecimal), then the residue table consists of

N_1	11011010 00001001 00000000	DA0900
N_2	01101101 00000100 10000000	6D0480
N_3	00110110 10000010 01000000	368240
N_4	00011011 01000001 00100000	1B4120
N_5	00001101 10100000 10010000	0DA090
N_6	00000110 11010000 01001000	06D048
N_7	00000011 01101000 00100100	036824
N_8	00000001 10110100 00010010	01B412
N_9	00000000 11011010 00001001	00DA09 $(= N)$.

This residue table can be constructed when the smart card is issued, because the values of P, Q and N are fixed when the card is issued.

The residue algorithm using the residue table is as follows;

Algorithm MOD :

Input B and N

Output $A = B \bmod N$

Step 1 Compare the length of B and N.

Step 1-1 if $|B|_2 < |N|_2$ then output B as A.

Step 1-2 if $|B|_2 = |N|_2$ then go to **Step 2**.

Step 1-3 Search i such that the first byte of N_i is greater than the first byte of B.

Step 1-4 Compute $B := B - N_i 2^{|B|_2 - |N_i|_2}$.

Step 1-5 Return **Step 1**.

Step 2 Compare B and N.

Step 2-1 if $B < N$ then output B as A.

Step 2-2 Compute $B := B - N$.

Step 2-3 Return **Step 2**.

3.4 Pre-computation

In this subsection, we will show the other technique for speeding up signature generation.

The signature generation procedure of ESIGN contains some computations independent of the message to be signed. Hence, these steps can be carried out before receiving a message to be signed. This technique dramatically reduces the processing time for signature generation.

The below steps are computed in the *pre-computation phase*.

- $X \in_R Z^*_{PQ}$,
- $F = X^K \bmod N$,
- $G = KX^{K-1} \bmod P$,
- $T = 1/G \bmod P$.

The steps which depend on M are as follows (in other words, these steps are computed in the *signature generation phase*):

- $W = \left\lceil \dfrac{H(M) - F}{PQ} \right\rceil$,
- $Y = WT \bmod P$,
- $S = X + YPQ$.

After the pre-computation, triplet (X, F, T) is stored in EEPROM. When message M is input, only three steps (computing W, Y, S) are needed to generate signature S. Note that each triplet (X, F, T) is used for each signature S.

Clearly, the pre-computation phase including the power and inversion operations is much more computationally demanding than the signature generation phase, which consists of only addition, multiplication and division operations.

3.5 Performance

This subsection introduces the experimental data of our implementation of ESIGN.

Table. 1 shows the memory size of the program and data area. This table was estimated assuming $|P|_{64} = 32$ (P's size is 32bytes), or $|N|_{64} = 96$. This sizes of P and N guarantee that the degree of security of our digital signature scheme is comparable to that of the RSA scheme with 64byte (512bit) modulus, assuming that the security of our scheme and the RSA scheme only depend on factoring each modulus (or N in our scheme). This table shows that our implementation satisfies the memory restriction of a typical smart card.

Table 1: Required memory size for ESIGN

Memory Area	Required Memory Size
RAM (bytes)	352
ROM (bytes)	2,882
EEPROM	
P (bytes)	32
Residue Table of P (bytes)	297
PQ (bytes)	64
Residue Table of PQ (bytes)	585
N (bytes)	96
Residue Table of N (bytes)	873
X, F, T (bytes)	192
Total (bytes)	2,139

Table. 2 lists the processing speeds, when $K = 4$[§]. When the size of P is 32bytes, the running time of pre-computation is just under 4 seconds. Therefore, a smart card can do pre-computation easily in any idling time such as a time waiting for a message to be input. Then, the digital signature can be generated within 0.2 seconds.

Note that signature verification is also fast (about 2 seconds); fast enough for most applications, although verification is usually executed in centers and terminals, not in smart cards.

[§]The recommended values of K are 4, 8, 16, 32, 64, and 128. When parameter K is one of these recommended values, the computation times for signature generation and pre-computation are at most twice as much as those of $K = 4$.

Our implementation uses a straightforward algorithm (or standard extended Euclidean algorithm) for computing the modular division (computing T from G and P). If we use a more efficient algorithm (e.g., [Knu], algorithm L and B, pp.321-339), then the speed of pre-computation will increase a few times, although the program size may increase.

Table 2: Performance of ESIGN

$\lvert P\rvert_{64}$	16	24	32	40
Card Issuing (sec)	0.02	0.04	0.07	0.10
Pre-Computation (sec)	0.67	1.86	3.88	7.28
Generating Signature (sec)	0.04	0.09	0.18	0.24
Verifying Signature (sec)	0.45	0.98	2.05	2.50

4 Identification Schemes

This section introduces two practical identification schemes based on ESIGN, which are at least as efficient as the Fiat-Shamir identification scheme [FS]. The two schemes are the three round version and the two round version. We omit the description of the identity-based versions [Sh] of these identification schemes, because we can easily modify these public-key file versions into the identity-based versions by using Kohnfelder's idea [De].

(1) Identification Scheme 1 (*three* round version)

Protocol:

Step 1 (Key generation and registration) Prover P generates ESIGN's secret key, p_P, q_P, and public key, n_P, and publishes the public key with his name.

Step 2 Prover P generates random numbers r, u, and calculates $x = f(r\|u)$, where f is a one-way function. P sends x to verifier V.

Step 3 V generates random number y and sends y to P.

Step 4 P calculates $m = r \oplus y$. P generates ESIGN's signature s of m using P's secret key, and sends (s, r, u) to V.

Step 5 V checks whether $x = f(r\|u)$ holds and whether s is a valid signature of m using P's public key.

(2) Identification Scheme 2 (*two* round version)

Protocol:

Step 1 (Key generation and registration) Same as **Step 1** of Scheme 1.

Step 2 V generates random number r and sends y to P.

Step 3 P generates random number u and calculates $m = f(r||u)$. P generates ESIGN's signature s of m using P's secret key, and sends (s, u) to V.

Step 4 V checks whether $m = f(r||u)$ holds and whether s is a valid signature of m using P's public key.

5 Conclusion

We have presented a practical digital signature implementation scheme for smart cards. We have shown that the signature generation time is just about 0.2 seconds. This is achieved using pre-computation and table look-up techniques. The required memory size is small enough for smart card implementation. The program size and the data size including the work area are at most 3Kbytes each.

Thus, we can conclude that our smart card digital signature scheme can be put to practical use, from the viewpoints of processing time and the required memory size.

Acknowledgment

The authors wish to thank Masahiko Iwata for his basic idea of the table look-up method.

References

[Br] E. F. Brickell, "A Survey of Hardware Implementation of RSA", in *Advances in Cryptology — CRYPTO '89*, Lecture Notes in Computer Science 435, Springer–Verlag, Berlin, pp.368–370 (1990).

[Da] D. W. Davies, "Applying the RSA Digital Signature to Electronic Mail", *Computer*, Vol.16, No.2, pp.55–62 (Feb., 1983).

[De] D. E. Denning, *Cryptography and Data Security*, Addison-Wesley, pp.170 (1982).

[DH] W. Diffie and M. E. Hellman, "New Directions in Cryptography", *IEEE Transactions on Information Theory*, Vol.IT-22, No.6, pp.644–654 (Nov., 1976).

[EGM] S. Even, O. Goldreich, and S. Micali, "On-line/Off-line Digital Signatures", in *Advances in Cryptology — CRYPTO '89*, Lecture Notes in Computer Science 435, Springer–Verlag, Berlin, pp.263–275 (1990).

[FS] A. Fiat and A. Shamir, "How to Prove Yourself: Practical Solutions to Identification and Signature Problems", in *Advances in Cryptology — CRYPTO '86*, Lecture Notes in Computer Science 263, Springer–Verlag, Berlin, pp.186–194 (1987).

[GQ] L. C. Guillou and J. J. Quisquater, "A Practical Zero-Knowledge Protocol Fitted to Security Microprocessors Minimizing both Transmission and Memory", in *Advances in Cryptology — EUROCRYPT '88*, Lecture Notes in Computer Science 330, Springer–Verlag, Berlin, pp.123–128 (1988).

[KH] S. Kawamura and K. Hirano, "A Fast Modular Arithmetic Algorithm Using a Residue Table", in *Advances in Cryptology — EUROCRYPT '88*, Lecture Notes in Computer Science 330, Springer–Verlag, Berlin, pp.246–250 (1988).

[Kno] H. Knobloch, "A Smart Card Implementation of the Fiat-Shamir Identification Scheme", in *Advances in Cryptology — EUROCRYPT '88*, Lecture Notes in Computer Science 330, Springer–Verlag, Berlin, pp.87–95 (1988).

[Knu] D. E. Knuth, *The Art of Computer Programming* 2nd Edition, Vol.2, *Semi-Numerical Algorithms*. Reading, Massachusetts: Addison-Wesley (1981).

[MOI] S. Miyaguchi, K. Ohta, and M. Iwata, "A 128-bit Hash Function (N-Hash)", *Proceedings of SECURICOM'90*, pp.123–137 (Mar., 1990).

[OhO] K. Ohta and T. Okamoto, "A Modification of the Fiat-Shamir Scheme", in *Advances in Cryptology — CRYPTO '88*, Lecture Notes in Computer Science 403, Springer–Verlag, Berlin, pp.232–243 (1990).

[Ok] T. Okamoto, "A Fast Signature Scheme Based on Congruential Polynomial Operations", *IEEE Transactions on Information Theory*, Vol.IT-36, No.1, pp.47–53 (Jan., 1990).

[OkS] T. Okamoto and A. Shiraishi, "A Digital Signature Scheme Based on Quadratic Inequalities", *Proceeding of Symposium on Security and Privacy*, IEEE, pp.123–132 (Apr., 1985).

[OnS] H. Ong and C. P. Schnorr, "A Fast Signature Generation with the Fiat-Shamir Scheme", in *Advances in Cryptology — EUROCRYPT '90*, Lecture Notes in Computer Science 473, Springer–Verlag, Berlin, pp.432–440 (1991).

[RSA] R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", *Communications of the ACM*, Vol.21, No.2, pp.120–126 (Feb., 1978).

[Sh] A. Shamir, "Identity-based Cryptosystems and Signature Schemes", *Advances in Cryptology — CRYPTO '84*, Lecture Notes in Computer Science 196, Springer–Verlag, Berlin, pp.47–53 (1985).