Memory Efficient Variants of Public-Key Schemes for Smart Card Applications

Adi Shamir

Applied Mathematics Department
The Weizmann Institute of Science
Rehovot 76100, Israel

Abstract

We propose new variants of the Rabin encryption scheme and the Fiat-Shamir identification scheme which require only a small fraction of the random access memory (RAM) required by the original schemes. The improved variants are provably as secure as the original variants, and can be implemented on standard smart cards with as few as 36 bytes of RAM without using dedicated coprocessors.

1 Introduction

Almost all the public key encryption and identification schemes proposed so far are based on modular multiplications with a modulus n which is the product of two secret primes p and q. To make the factorization of n difficult, it is necessary to use very large numbers. The minimum recommended size of n is currently 512 bits, but due to the explosive growth of computing power available to cryptanalysts, this minimum size is likely to increase to 1024 bits (and to even larger values for high security applications).

In many communication and access control applications, it is desirable to use smart cards to carry out the sensitive computations. Among the many reasons for such a choice are the small physical size, the portability, the convenience of non-volatile memory, and the security offered by a single chip computer embedded in a plastic card. Millions of smart cards are used each year to make bank cards more secure, to control access to pay-TV, to carry billing information in cellular telephones, etc.

The biggest limitation in today's smart cards is the small amount of random access memory (RAM) available in the card. The most popular smart card chip made by Motorola has 36 bytes of RAM, and the most popular smart card chip made by Thomson has 44 bytes of RAM. This should be compared with the 4 million bytes of RAM available in a typical personal computer.

Some algorithms can make use of other types of memory: A typical smart card contains several thousand bytes of ROM and several thousand bytes of EEPROM. ROM is unchangeable, and typically stores the program which controls the operation of the smart card. EEPROM is changeable, but writing into it is about one thousand time slower than writing into RAM and the number of times a bit can be rewritten cannot exceed 10,000. It is thus possible to use EEPROM to store slowly changing data such as cryptographic keys or the details of financial transactions, but not as a RAM substitute for intermediate values in a long computation.

In this paper, we show how to modify in a provably secure way two of the most popular public key schemes (the Rabin encryption scheme and the Fiat-Shamir identification scheme) in order to make them suitable for smart cards with severely limited RAM's. The standard implementations of the original schemes require quadratic time and linear space (as a function of the modulus size). The modified variants require quadratic time but only logarithmic space, and thus even the simplest 36 byte smart card can handle moduli n with thousands of bits without any difficulty.

2 Randomized Multiplication

We first consider the basic operation of multiplying two large numbers x and y which are already stored in the smart card (e.g., in its EEPROM). The result $z = x \cdot y$ cannot fit in the small RAM, but if it is the final result computed by the smart card, its successive bytes can be sent out (rather than stored) as soon as they are generated.

The classical method for multiplying two $O(k)$-byte numbers in $O(k^2)$ time with $O(log(k))$ workspace is to use convolution: Start with $c = 0$. To compute the i-th byte of the result z for $i = 0, 1, 2, \dots, k$, compute $t = c + \sum_{j=0}^{k} x_j \cdot y_{i-j}$ for$j = 0, 1, 2, \dots, i$, send the least significant byte of t as z_i, and use the value of the other bytes of t as the new carry c. Note that for any x and y with up to half a million bits, t fits into 4 bytes, and thus the algorithm can be easily implemented even on a smart card with 36 bytes of RAM.

Next we consider the problem of computing $z = x \cdot y \pmod{n}$. By definition, $z = x \cdot y - w \cdot n$ where $w = \lfloor x \cdot y / n \rfloor$ (i.e., w is $x \cdot y / n$ truncated to the largest integer below it). Since we cannot store z, it is not obvious how to carry out this division operation. We can try to generate the successive bytes of z by the convolution method, but in the division operation we need these bytes from left to right, whereas the convolution computes them from right to left. We are thus forced to recalculate each byte of z a large number of times, and the cryptographic scheme becomes unacceptably slow.

We solve this problem by replacing the modular multiplication operation $z = x \cdot y - w \cdot n$ where $w = \lfloor x \cdot y/n \rfloor$ by a new randomized multiplication operation $z' = x \cdot y + r \cdot n$ where r is a randomly chosen integer in a suitable range $[0, b]$. Such a z' can be easily computed by the following double convolution process:

1. Set $c = 0$.

2. For $i = 0, 1, \ldots, k$ compute $t = c + \sum x_j \cdot y_{i-j} + \sum r_m \cdot n_{i-m}$, send the low-order byte of t as z_i, and set c to the number represented by the other bytes of t.

Clearly, the value of z can be recovered from the value of z' by reducing z' modulo n, and thus there is no loss of information in sending z' instead of z. We now show that when b is a large enough public bound, there is no loss of cryptographic security in sending z' instead of z. Assume that there exists an attack against the cryptographic scheme which uses the knowledge of $z' = x \cdot y + r \cdot n$ for a random r in $[0, b]$. These values define a probability space P'. We now show that such an attack can also be based on the deterministic value of $z = x \cdot y - w \cdot n$. Given z, the cryptanalyst can easily compute by himself another probability space P'' defined by the values $z'' = z + u \cdot n$ for a random u in $[0, b]$. By definition, $z'' = x \cdot y + (u - w) \cdot n$. The only difference between P' and P'' is that the range of coefficients of n is shifted by the unknown quantity w from $[0, b]$ to $[-w, b - w]$. The probability that a random point in one range will fall outside the other is w/b. If x and y are in $[0, n]$, and b is much larger than n, then this probability is negligible (e.g., when b is a power of 2 which is 64 bits longer than n, this probability is less than 2^{-64}). Since P' and P'' are statistically indistinguishable, any cryptanalytic attack will be equally successful over the two probability spaces.

The only disadvantage of randomized multiplication with respect to modular multiplication is that the transmitted result is about twice as long. However, this adds only a negligible communication delay, and the receipient (which is usually a powerful PC or a network server) can immediately change z' to z before storing or processing it further.

In the next two sections we describe how to use randomized multiplications in order to obtain space-efficient variants of the Rabin and Fiat-Shamir cryptographic schemes.

3 A Space-Efficient Variant of the Rabin Encryption Scheme

Rabin's public key scheme is used to establish a common secret key K between two parties, which can then be used to encrypt the actual message with a standard cryptosystem such as DES. K is usually the low order bits (56 in the case of DES) of a long number x in the range $[0, n]$ chosen randomly by the sender. The sender then computes and sends $z = x \cdot x \pmod{n}$, and the receipient uses his knowledge of the factorization of n in order to compute the modular square root of $z \pmod{n}$. A slight complication is that z yields four possible K's, but this can be easily solved by adding some redundancy.

In the proposed new variant, K is sent by computing the randomized multiplication $z' = x \cdot x + r \cdot n$ instead of the modular multiplication $z = x \cdot x \pmod{n}$. In a typical implementation, n (which is the product of two large primes p and q) is kept in EEPROM. It is usually the public key of the organization which issues the cards to its employees and customers, and even small cards with one kilobyte of EEPROM can store an 8000 bit modulus. The numbers x and r are pseudo randomly generated from random secret seeds s_x and s_r (which are loaded into EEPROM when the card is issued), a session counter v (which is incremented in EEPROM at the beginning of each communication session), and a byte index j. A convenient way of achieving this is to hash s_x, v, and j into the j-th byte of x in session v, and to hash s_r, v, and j into the j-th byte of r in session v. In this way it is possible to access individual bytes of x and r in any order without storing x and r anywhere, and thus it is possible to compute and send out the successive bytes of z' by the double convolution method even when n has thousands of bits and the card has only 36 bytes of RAM.

4 A Space-Efficient Variant of the Fiat-Shamir Identification Scheme

In this section we describe a space-efficient variant of the Fiat-Shamir identification scheme which makes it possible to implement it on smart cards with very small RAM's.

In the original Fiat-Shamir identification scheme, the smart card (known as the prover) contains in its EEPROM a public modulus $n = p \cdot q$ and a secret number c. The other party (known as the verifier) knows n and

$d = c \cdot c \pmod{n}$. The smart card proves its identity to the verifier by using a zero knowledge protocol to demonstrate its knowledge of c. The proof consists of the following steps:

1. The prover chooses a random number x, and sends $z = x \cdot x \pmod{n}$ to the verifier.
2. The verifier sends a random bit to the prover.
3. Based on this bit, the prover sends either x or $x \cdot c \pmod{n}$ to the verifier.
4. Based on this bit, the verifier checks that the square of the received number is either $z \pmod{n}$ or $z \cdot d \pmod{n}$.
5. Steps 1–4 are repeated several times to reduce the probability of cheating.

In the new space-efficient variant of the Fiat-Shamir scheme, the prover performs the same steps, but replaces the modular multiplications $x \cdot x \pmod{n}$ and $x \cdot c \pmod{n}$ by the randomized multiplications $x \cdot x + r \cdot n$ and $x \cdot c + t \cdot n$ for pseudo-random r and t in $[0, b]$, where b is substantially larger than n (e.g., by at least 64 bits). As demonstrated in Section 2, these operations can be carried out with very small RAM's.

Remark: After the presentation of this paper at Eurocrypt 94, David Naccache pointed out that a related idea appeared in his European Patent Application 91402958.2, submitted on November 5-th 1991. In his patent application he also adds random multiples of n to various values, but his goal is to reduce the computation time by a constant factor, whereas our goal is to reduce the RAM space from linear to logarithmic. Another difference is that his variant is not proven to be as secure as the original variant, and some of its possible implementations are in fact breakable.