Montgomery Exponentiation Needs No Final Subtractions

Colin D. Walter
Computation Department, UMIST
PO Box 88, Sackville Street, Manchester M60 1QD, UK
www.co.umist.ac.uk

Abstract

Montgomery's modular multiplication algorithm is commonly used in implementations of the RSA cryptosystem. We observe that there is no need for extra cleaning up at the end of an exponentiation if the method is set up in the right way.

Key Words: Cryptography, RSA cryptosystem, Montgomery modular multiplication.

1 Introduction

The RSA encryption function [6] uses a public modulus M, usually of up to 1024 bits, and two keys d and e, at least one of which is private, with the property that $A^{de} \equiv A \bmod M$. Message blocks A satisfying $0 \le A < M$ are encrypted to $C = A^e \bmod M$ and decrypted by $A = C^d \bmod M$. The computation of $A^e \bmod M$ consists of two main processes: modular multiplication and exponentiation. The constituent modular reduction steps on partial results S normally require a comparison of S with M which, in the worst case, means that every bit of both long numbers needs to be examined in turn. Such potentially expensive comparisons need to be done even if no subtraction is actually required. This letter will show that use of Montgomery's modular multiplication algorithm [5] enables *every* such comparison to be avoided. It develops further a remark by Blum and Paar [1]. The methods are applicable to all implementations, whether in software or hardware. As well as saving computation time, avoiding such comparisons is important in preventing the success of timing attacks on the cryptosystem [4].

2 Montgomery's Algorithm

Suppose all numbers are represented with base (or radix) r which is a power of 2, say $r = 2^k$, and let n be an upper bound on the number of digits needed for any number encountered. Then every number X has a representation of the form $X = \sum_{i=0}^{n-1} x_i r^i$ where, for non-redundant systems, the ith digit x_i satisfies $0 \le x_i < r$. The classical modular multiplication algorithm for $(A\times B)$ mod M simply takes the normal method of multiplication, which accumulates digit products $a_i\times B$, and interleaves modular reductions to keep the result below M. Peter Montgomery [5] has provided a variation of this algorithm in which the multiplier digits are consumed in the reverse order and no full length comparisons are required for the modular reductions:

MONTGOMERY'S MODULAR MULTIPLICATION ALGORITHM

```
{ Pre-condition: M prime to r and A non-redundant}
S := 0 ;
For i := 0 to n-1 do
Begin
   qi := (s0+aib0)(-m0^-1) mod r ;
   S  := (S + ai×B + qi×M) div r ;
   { Invariant: 0 ≤ S < M+B }
End ;
{ Post-condition: Sr^n = A×B + Q×M }
```

In RSA the modulus is a product of two large primes and so prime to the radix r. Hence there is a residue m_0^{-1} mod r which satisfies $m_0\times m_0^{-1} \equiv 1$ mod r. The digit q_i is chosen so that $S+a_i\times B+q_i\times M$ is exactly divisible by r. If we define $A_i = \sum_{j=0}^{i} a_j r^j$ and Q_i analogously then $A_i = A_{i-1}+a_i r^i$ and $A_n = A$. By induction, the value of S at the end of the ith iteration is easily shown to satisfy $r^{i+1}S = A_i\times B+Q_i\times M$ because the division is exact. Hence the post-condition holds. Moreover, the bound on the size of the digits a_i enables the loop invariant to be established also by induction.

Some implementations may make use of redundant representations in which the digits have a wider range than 0 .. $r-1$. However, because the digits of A are consumed in ascending order, they can be converted on-line into the standard representation for A. Thus the algorithm can treat any redundancy in A. Redundancy in the other numbers is immaterial to the present argument.

3 Exponentiation

In an encryption, the extra power of r factor in the output S is easily cleared up by minor processing before and after the exponentiation [2], [3]. We associate with every number its *Montgomery class* mod M, namely

$$\overline{A} \;\equiv\; r^n A \bmod M$$

Then, if $\overline{\times}$ denotes Montgomery modular multiplication, the *Montgomery product* of $\overline{A}$ and $\overline{B}$ is $\overline{A}\,\overline{\times}\,\overline{B} \;\equiv\; \overline{A}\,\overline{B}\,r^{-n} \;\equiv\; ABr^n \;\equiv\; \overline{AB} \bmod M$. Hence, using $\overline{\times}$ rather than $\times$ in an exponentiation algorithm is going to produce $\overline{A^e}$ from $\overline{A}$. The initial class $\overline{A}$ is normally formed as a Montgomery product from A and the pre-computed value

$$R = \overline{r^n} \equiv r^{2n} \bmod M$$

by computing

$$A\,\overline{\times}\,R \;\equiv\; Ar^n \;\equiv\; \overline{A} \bmod M.$$

Finally, removal of the extra power of r from $\overline{A^e}$ is also done by a Montgomery multiplication: A^e mod M is obtained from

$$\overline{A^e}\;\overline{\times}\,1 \equiv A^e \bmod M.$$

4 Bounds on the I/O

Throughout the exponentiation, outputs from multiplications are re-used as inputs. So it is important to ensure these numbers remain bounded. In particular, we will show $S < 2M$ is maintainable for all outputs S. Assume that n is large enough for $2M < r^{n-1}$ to hold and that the inputs A, B to a Montgomery multiplication both satisfy the bound, i.e. $A < 2M$ and $B < 2M$. Then $a_{n-1} = 0$. Hence, the bound $S < M+B$ at the end of the second last loop iteration yields $S < M+r^{-1}B$ on the final round, from which $S < 2M$ (as $r \geq 2$). Therefore, as both R and the initial message A for encryption should be below the bound $2M$, the final output of the exponentiation should also satisfy this bound.

Now consider the final scaling by 1 to remove the unwanted power of r from $\overline{A^e}$. The post-condition of this modular multiplication is $Sr^n = \overline{A^e} + QM$. Here Q can have a maximum value of r^n-1 arising from all its digits being $r-1$. So the bound $\overline{A^e} < 2M$ leads to $Sr^n < (r^n+1)M$ and thence to $S \le M$ because S is an integer and $r^{-n}M < 1$. Hence a final subtraction to obtain an output $S < M$ is *only* necessary if $S = M$, i.e. when $\overline{A^e} \equiv 0 \bmod M$, that is, for $A \equiv 0 \bmod M$. However, A is a plaintext or ciphertext message and hence, by definition, less than M. The only possibility is then that $A = 0$. But $A = 0$ clearly leads to *all* numbers being identically 0 throughout the exponentiation. In particular, the final output is 0 and does not require any extra subtraction. Thus, in no circumstances does the output A^e from the exponentiation need any further modular adjustment to obtain a least non-negative residue mod M.

4 Conclusion

We have considered implementations of the RSA cryptosystem which use solely Montgomery's modular multiplication algorithm and shown that under standard, easily met, inexpensive conditions, the total encryption process *never* needs any extra subtractions to produce output in the correct range.

References

[1] T. Blum & C. Paar, "Montgomery Modular Exponentiation on Reconfigurable Hardware", *Proc. 14th IEEE Symp. on Computer Arithmetic*, Adelaide, 14-16 April, 1999, pp. 70-77.

[2] S. E. Eldridge, "A Faster Modular Multiplication Algorithm", *Intern. J. Computer Math.*, vol. 40, 1991, pp. 63-68.

[3] S. E. Eldridge & C. D. Walter, "Hardware Implementation of Montgomery's Modular Multiplication Algorithm", *IEEE Trans. Comp.*, vol. 42, 1993, pp. 693-699.

[4] P. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems", *Advances in Cryptology, Proc Crypto 96*, Lecture Notes in Computer Science, vol. 1109, N. Koblitz editor, Springer-Verlag, 1996, pp 104-113.

[5] P. L. Montgomery, "Modular Multiplication without Trial Division", *Math. Computation*, vol. 44, 1985, pp. 519-521.

[6] R. L. Rivest, A. Shamir & L. Adleman, "A Method for obtaining Digital Signatures and Public-Key Cryptosystems", *Comm. ACM*, vol. 21, 1978, pp. 120-126.