Increasing the Bitlength of a Crypto-Coprocessor

Wen-Chun Yang
School of Electrical Engineering and Computer Science
Oregon State University, Corvallis, OR 97331
E-mail: {yangwe}@ece.orst.edu

Abstract— **the purpose of this paper is to present the demonstrations for the algorithms proposed in [1]. The techniques presented in [1] increase the virtual bit-length of the crypto-coprocessors from both hardware and software point of views. While demonstrating the methods in [1], the algorithms and the comparisons would also be introduced in the rest of this paper.**

Key Words: Public-key cryptosystems, Arithmetical coprocessor, Hardware architecture, Modular multiplication, Hardware/Software codesign.

I. Introduction

Due to the need for using public-key cryptosystems in an efficient way, finding how to increase the bit-length of the co-processors has become one of the top priorities for the computer architects and the cryptographic engineers. This has become an open solution for the society, and to this day, no correct solutions have been published to the world.

One of the reasons for increasing the bit-length of a crypto-coprocessor is because of the large integer arithmetic, which is essential for the public-key cryptography. Many modular multiplication algorithms have been studied in order to speed up the arithmetic process [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17].

However, the high popularity of using portable devices has also become the other motivation for increasing the bit-length due to the fact that the low-power hardware devices do not provide sufficient bit-lengths.

A 2048-bit RSA can not be handled efficiently on a 1024-bit device. Therefore, those two concerns have become big issues for the industry. One of the solutions is using the Chinese Reminder Theorem for the RSA signature [4]. In order to keep the RSA verification simple, the fourth Fermat number is used as public exponent, which efficiently reduces the 2048-bit modular multiplications to 1024-bit modular multiplication [6] [7] [13].

In this paper, I will focus on the accuracy for the proposed algorithms in [1]. The rest of the paper is organized as follows. Section II explains the preliminaries. Section III defines the comparison of the doubling algorithm for the proposed algorithms in [1]. Section IV concludes this paper.

II. Prelimineries

Doubling Algorithm

A doubling algorithm is used when the data size is expanded because it minimizes the amount of copying that must be done.

The Instructions MultMod and MultModInitn

The following definition is the used for the usual modutlar multiplication [1].

Definition 1.
For numbers A, B, and N, $N > 0$, the MultMod instruction is defined as
$R = MultMod(A, B, N)$
with
$R := (A * B) mod N.$
The following extension of the modular multiplication is already the feathers of todays crypto-coprocessors [1].

Definition 2.
For a fixed integer n and numbers A, B, C and N, $N > 0$, the MultModInitn instruction is defined as
$R = MultModInitn(A, B, C, N)$
with
$R := (A * B + C * 2^n) mod N.$

Definition 3.
The following definition is a natural extension of the usual modular multiplication [1]. For a fixed integer n and numbers A, B, and N, $N > 0$, the MultModDiv instruction is defined as
$(Q, R) = MultModDiv(A, B, N)$
with
$Q := \lfloor \frac{A*B}{N} \rfloor$
$R := (A * B) - Q * N.$

Definition 4.
For a fixed integer n and numbers A, B, C, and N, $N > 0$, the MultModDivInitn instruction is defined as
$(Q, R) = MultModDivInitn(A, B, C, N)$
with
$Q := \lfloor \frac{A*B+C*2^n}{N} \rfloor$
$R := (A * B + C * 2n) - Q * N.$

III. The Doubling Algorithm

A. Modular Multiplicatoin without Initialization

Theorem 1.
There exists an algorithm to compute $A * B$ mod N using seven MultModDiv instructions of length n, provided that $2^{2n-1} \leq N < 2^{2n}$ and $0 \leq A, B < N$.
proof: we will first present the algorithm.

Blasic Doubling Algorithm
Input:
$N = Nt2^n + Nb$ with $0 \le Nb < 2^n$.
$A = At2^n + Ab$ with $0 \le Ab < 2^n$.
$B = Bt2^n + Bb$ with $0 \le Bb < 2^n$.

```
1:  $(Q^1, R^1) = MultModDiv(Bt, 2^n, Nt)$
2:  $(Q^2, R^2) = MultModDiv(Q^1, Nb, 2^n)$
3:  $(Q^3, R^3) = MultModDiv(At, R^1 - Q^2 + Bb, Nt)$
4:  $(Q^4, R^4) = MultModDiv(Ab, Bt, Nt)$
5:  $(Q^5, R^5) = MultModDiv(Q^3 + Q^4, Nb, 2^n)$
6:  $(Q^6, R^6) = MultModDiv(At, R^2, 2^n)$
7:  $(Q^7, R^7) = MultModDiv(Ab, Bb, 2^n)$
8:  $Q := (R^3 + R^4 - Q^5 - Q^6 + Q^7)$
9:  $R := (R^7 - R^6 - R^5)$
10: make final reduction on $(Q * 2^n + R)$
```
Output: $Q * 2^n + R$

From the Input equations, we will get:
$Q^1 := \lfloor \frac{Bt*Z}{Nt} \rfloor$
$R^1 := (Bt * Z) - Q^1 * Nt$
$Q^2 := \lfloor \frac{Q^1*Nb}{Z} \rfloor$
$R^2 := (Q^1 * Nb) - Q^2 * Z$
$Q^3 := \lfloor \frac{At*R^1-Q^2+Bb}{Nt} \rfloor$
$R^3 := (At * R^1 - Q^2 + Bb) - Q^3 * Nt$
$Q^4 := \lfloor \frac{Ab*Bt}{Nt} \rfloor$
$R^4 := (Ab * Bt) - Q^4 * Nt$
$Q^5 := \lfloor \frac{(Q^3+Q^4)*Nb}{Z} \rfloor$
$R^5 := ((Q^3 + Q^4) * Nb) - Q^5 * Z$
$Q^6 := \lfloor \frac{At*R^2}{Z} \rfloor$
$R^6 := (At * R^2) - Q^6 * Z$
$Q^7 := \lfloor \frac{Ab*Bb}{Z} \rfloor$
$R^7 := (Ab * Bb) - Q^7 * Z$

The author in [1] tried to prove that
$(R^3 + R^4 - Q^5 - Q^6 + Q^7) * 2^n + (R^7 - R^6 - R^5)$
is indeed congruent to $A * B$ mod N. This demonstration can be easily shown as follows, where $Z = 2^n$ as abbreviation.

$(AtZ + Ab) * (BtZ + Bb)$
$= AtBtZZ + AtBbZ + AbBtZ + AbBb$
$= At(R^1 + Q^1Nt)Z + AtBbZ + AbBtZ + AbBb$
$= AtR^1Z + AtQ^1(-Nb) + AtBbZ + AbBtZ + AbBb$
$= (R^3 + AtQ^2 - AtBb + Q^3Nt)Z - AtNbQ^1 + AtBbZ + AbBtZ + AbBb$
$= R^3Z + AtQ^2Z - AtBbZ + Q^3NtZ - AtNbQ^1 + AtBbZ + AbBtZ + AbBb$
$= R^3Z + AtQ^2Z + Q^3NtZ - AtNbQ^1 + AbBtZ + AbBb$
$= R^3Z + AtQ^2Z + Q^3NtZ - AtNbQ^1 + (R^4 + Q^4Nt)Z + AbBb$
$= (R^3 + R^4)Z + ((Q^3 + Q^4)NtZ + AtQ^2Z - AtNbQ^1 + AbBb$
$= (R^3 + R^4)Z + ((Q^3 + Q^4)NtZ + AtQ^2Z - At(R^2 + Q^2Z) + AbBb$
$= (R^3 + R^4)Z + ((Q^3 + Q^4)NtZ - AtR^2 + AbBb$
$= (R^3 + R^4)Z + ((Q^3 + Q^4)NtZ - (R^6 + Q^6Z) + AbBb$
$= (R^3 + R^4 - Q^6)Z + ((Q^3 + Q^4)NtZ - R^6 + AbBb$
$= (R^3 + R^4 - Q^6)Z + ((Q^3 + Q^4)NtZ - R^6 + (R^7 + Q^7Z)$
$= (R^3 + R^4 + Q^7 - Q^6)Z + ((Q^3 + Q^4)(-Nb) + (R^7 - R^6)$
$= (R^3 + R^4 + Q^7 - Q^6)Z - ((R^5 + Q^5Z) + (R^7 - R^6)$
$= (R^3 + R^4 + Q^5 - Q^6 + Q^7)Z + (R^7 - R^6 - R^5) modN$

However, this demonstration is based on the fact that $NtZ = -Nb$ mod N. Otherwise, it would fail from the third to the fourth equation. Using the " associative law " to switch the parameters of the input equations completes the proof. Note that if the given module N has an odd bitlength, then one has to compute with 2^n.

B. Modular Multiplicatoin with Initialization
Theorem 2.
There exists an algorithm to compute $A * B$ mod N using five MultModDiv instructions and one MultModDivInitn instruction of length n, provided that $2^{2n-1} \le N < 2^{2n}$ and $0 \le A, B < N$.
proof: we will present the algorithm.

Basic Doubling Algorithm
Input:
$N = Nt2^n + Nb$ with $0 \le Nb < 2^n$.
$A = At2^n + Ab$ with $0 \le Ab < 2^n$.
$B = Bt2^n + Bb$ with $0 \le Bb < 2^n$.

```
1:  $(Q^1, R^1) = MultModDiv(At, Bt, Nt)$
2:  $(Q^2, R^2) = MultModDivInitn(Nb, -Q^1, R^1, Nt)$
3:  $(Q^3, R^3) = MultModDiv(At, Bb, Nt)$
4:  $(Q^4, R^4) = MultModDiv(Ab, Bt, Nt)$
5:  $(Q^5, R^5) = MultModDiv(Ab, Bb, 2^n)$
6:  $(Q^6, R^6) = MultModDiv(Q^2 + Q^3 + Q^3, Nb, 2^n)$
7:  $Q := (R^2 + R^3 + R^4 + Q^5 - Q^6)$
8:  $R := (R^5 - R^6)$
9:  make final reduction on $(Q * 2^n + R)$
```
Output: $Q * 2^n + R$

From the Input equations, we will get:
$Q^1 := \lfloor \frac{At*Bt}{Nt} \rfloor$
$R^1 := (At * Bt) - Q^1 * Nt$
$Q^2 := \lfloor \frac{Nb*(-Q^1+R^1*2^n)}{Nt} \rfloor$
$R^2 := (Nb * (-Q^1 + R^1 * 2^n)) - Q^2 * Nt$
$Q^3 := \lfloor \frac{At*Bb}{Nt} \rfloor$
$R^3 := (At * Bb) - Q^3 * Nt$
$Q^4 := \lfloor \frac{Ab*Bt}{Nt} \rfloor$
$R^4 := (Ab * Bt) - Q^4 * Nt$
$Q^5 := \lfloor \frac{Ab*Bb}{Z} \rfloor$
$R^5 := (Ab * Bb) - Q^5 * Z$
$Q^6 := \lfloor \frac{(Q^2+Q^3+Q^4)*Nb}{Z} \rfloor$
$R^6 := ((Q^2 + Q^3 + Q^4) * Nb) - Q^6 * Z$

At this point, the author in [1] tried to prove that $(R^2 + R^3 + R^4 + Q^5 - Q^6) * 2^n + (R^5 - R^6)$ is indeed congruent to $A * B$ mod N. This demonstration can be easily shown as follows, where $Z = 2^n$ as abbreviation.

$(AtZ + Ab) * (BtZ + Bb)$
$= AtBtZZ + AtBbZ + AbBtZ + AbBb$
$= ((Q^1Nt + R^1)ZZ + AtBbZ + AbBtZ + AbBb$
$= (R^1Z - Q^1Nb)Z + AtBbZ + AbBtZ + AbBb$
$= (Q^2Nt + R^2)Z + AtBbZ + AbBtZ + AbBb$
$= (R^2Z - Q^2Nb) + AtBbZ + AbBtZ + AbBb$
$= (R^2Z - Q^2Nb) + (Q^3Nt + R^3)Z + AbBtZ + AbBb$
$= (R^2Z - Q^2Nb) + (Q^3Nt + R^3)Z + (Q^4Nt + R^4)Z + AbBb$
$= (R^2Z - Q^2Nb) + (Q^3Nt + R^3)Z + (Q^4Nt + R^4)Z + (Q^5Z + R^5)$
$= ((R^2 + R^3 + R^4 + Q^5)Z + ((Q^2 + Q^3 + Q^4)(-Nb) + R^5))modN$

After going through the demonstration, what I found from the proposed algorithm is that the value of the parameters in the equation for Q^6 and R^6 should be changed to
$(Q^6, R^6) := MultModDiv(Q^2 + Q^3 + Q^4, Nb, 2^n)$.
The author in [1] proposed this instruction as
$(Q^6, R^6) := MultModDiv(Q^2 + Q^3 + Q^3, Nb, 2^n)$.
which will not get the final result ($A * B$ mod N) as what we need to prove. Therefore, the proposing algorithm is listed as follows.

New Proposing Basic Doubling Algorithm
Input:
$N = Nt2^n + Nb$ with $0 \leq Nb < 2^n$.
$A = At2^n + Ab$ with $0 \leq Ab < 2^n$.
$B = Bt2^n + Bb$ with $0 \leq Bb < 2^n$.

```
1: $(Q^1, R^1) = MultModDiv(At, Bt, Nt)$
2: $(Q^2, R^2) = MultModDivInitn(Nb, -Q^1, R^1, Nt)$
3: $(Q^3, R^3) = MultModDiv(At, Bb, Nt)$
4: $(Q^4, R^4) = MultModDiv(Ab, Bt, Nt)$
5: $(Q^5, R^5) = MultModDiv(Ab, Bb, 2^n)$
6: $(Q^6, R^6) = MultModDiv(Q^2 + Q^3 + Q^4, Nb, 2^n)$
7: $Q := (R^2 + R^3 + R^4 + Q^5 - Q^6)$
8: $R := (R^5 - R^6)$
9: make final reduction on $(Q * 2^n + R)$
```
Output: $Q * 2^n + R$

From the Input equations, we will get:
$[1]Q^1 := \lfloor \frac{At*Bt}{Nt} \rfloor$
$R^1 := (At * Bt) - Q^1 * Nt$
$Q^2 := \lfloor \frac{Nb*(-Q^1+R^1*2^n)}{Nt} \rfloor$
$R^2 := (Nb * (-Q^1 + R^1 * 2^n)) - Q^2 * Nt$
$Q^3 := \lfloor \frac{At*Bb}{Nt} \rfloor$
$R^3 := (At * Bb) - Q^3 * Nt$
$Q^4 := \lfloor \frac{Ab*Bt}{Nt} \rfloor$
$R^4 := (Ab * Bt) - Q^4 * Nt$
$Q^5 := \lfloor \frac{Ab*Bb}{Z} \rfloor$
$R^5 := (Ab * Bb) - Q^5 * Z$
$Q^6 := \lfloor \frac{(Q^2+Q^3+Q^4)*Nb}{Z} \rfloor$
$R^6 := ((Q^2 + Q^3 + Q^4) * Nb) - Q^6 * Z$

The demonstration for the new proposing algorithm would be:
$(AtZ + Ab) * (BtZ + Bb)$
$= AtBtZZ + AtBbZ + AbBtZ + AbBb$
$= (R^1 + Q^1Nt)ZZ + AtBbZ + AbBtZ + AbBb$
$= (R^1Z + Q^1(-Nb))Z + AtBbZ + AbBtZ + AbBb$
$= (R^1Z - Q^1Nb)Z + AtBbZ + AbBtZ + AbBb$
$= (R^2 - Q^2Nt)Z + AtBbZ + AbBtZ + AbBb$
$= (R^2Z - Q^2Nt)Z + (R^3 + Q^3Nt)Z + AbBtZ + AbBb$
$= (R^2Z - Q^2Nt)Z + (R^3 + Q^3Nt)Z + ((R^4 + Q^4Nt))Z + AbBb$
$= (R^2 + R^3 + R^4)Z + (Q^4 + Q^3 - Q^2)NtZ + AbBb$
$= (R^2 + R^3 + R^4)Z + (Q^4 + Q^3 - Q^2)NtZ + (R^5 + Q^5Z)$
$= (R^2 + R^3 + R^4 + Q^5)Z + (Q^4 + Q^3 - Q^2)NtZ + R^5$
$= (R^2 + R^3 + R^4 + Q^5)Z + (Q^4 + Q^3 - Q^2)(-Nb) + R^5$
$= (R^2 + R^3 + R^4 + Q^5)Z - (Q^2 + Q^3 + Q^4)Nb + R^5$
$= (R^2 + R^3 + R^4 + Q^5)Z - (R^6 + Q^6Z) + R^5$
$= ((R^2 + R^3 + R^4 + Q^5 - Q^6)Z + (R^5 - R^6))modN$

The changed algorithm is also based on the fact that $NtZ = -Nb$ mod N. However, the final proving shows the proposing algorithm from this paper will get the corrrect result, which completes the proof.

IV. Conclusion

In this paper, I demonstrated two basic doubling algorithm equations, which can efficiently compute the modular multiplications. Using the instructions MultModDiv and MultModDivInitn can improve algorithms and the results presented in [13]. However, there might be a typing mistake or different approach in [1] for the instruction MultModDivInitn, which actually does not result better than the one in [13]. However, the software and hardware realizations that use this proposed instruction in [13] can still present fast and better solutions for modular multiplication. In addition, the new change proposed in this paper will change the whole result and really make a better result compared with the one presented in [13], which is the goal for this paper to demonstrate the correctness of the algorithms proposed in [13].

References

[1] W. Fischer and J.-P. Seifert, "Increasing the bitlength of a crypto-coprocessor," *Proc. of CHES'02*, vol. 2523, pp. 71–81, 2003.

[2] P. Barret, "Implementing the rivest, shamir and adleman public-key encryption algorithm on a standard digital signal processor," *Proc. of CRYPTO'86*, vol. 263, pp. 311–323, 1987.

[3] S. Cavallar et alii, "Factoring a 512 bit rsa modulus," *Proc. of EUROCRYPT'00*, vol. 1807, pp. 1–19, 2000.

[4] J.-J. Quisquater C. Couvreur, "Fast decipherment algorithm for rsa public-key cryptosystem," *Electronics Letters*, vol. 18, no. 21, pp. 905–907, 1982.

[5] J.-J. Quisquater J.-F. Dhem, "Recent results on modular multiplication for smart cards," *Proc. of CARDES'98*, vol. 1820, pp. 336–352, 1988.

[6] P.Pailler H. Handschuh, "Smart card crypto-cpprocessors for public=key cryptography," *CryptoBytes*, vol. 4, no. 1, pp. 6–11, 1998.

[7] P. Pailler H. Handschur, "Smart card crypto-coprocessors for public-key cryptography," *Proc. of CARDIS'98*, vol. 1820, pp. 372–379, 1998.

[8] D. E. Knuth, vol. 2 of *Reading MA*, Addison-Wesley, 3rd edition, 1999.

[9] S. Vanstone A. J. Menezes, P. van Oorschot, New York, 1997.

[10] P. L. Montgomery, "Modular multiplication without trial division," *Mathematics of Computation*, vol. 44, pp. 519–521, 1985.

[11] D. M'Raihi D. Naccache, "Arithmetic co-processors for public-key cryptography: The state of the art," *IEEE Micro*, pp. 14–24, 1996.

[12] J. Omura, "A public key cell design for smart card chips," *Proc. of IT Workshop*, pp. 27–30, 1990.

[13] P. Pailler, "Low-cost double size modular exponentiation or how to stretch your crypto-coprocessor," vol. 1560, pp. 223–234, 1999.

[14] J.-J. Quisquater, "Encoding system according to the so-called rsa method, by means of a microcontroller and arrangement implementing this sytem," US Patent Nr. 5,166,979, November 24, 1992.

[15] H. Sedlak, "The rsa cryptographic processor: The first high speed one-chip solution," *Proc. of EUROCRYPT'87*, vol. 293, pp. 95–105, 1998.

[16] J.-J. Quisquater D. de Waleffe, "Corsair, a smart card for public-key cryptosystem," *Proc. of CRYPTO'90*, vol. 537, pp. 503–513, 1990.

[17] C. Walter, "Techniques for the hardware implementation of modular multiplication," *Proc. of 2nd IMACS Internet. Conf. on Circuits, Systems and Computers*, vol. 2, pp. 945–949, 1998.