C-013

冗長二進数を用いた RSA 暗号回路

Circuit for RSA cryptographic using Redundant Binary System

山本 和正†　　小柳 滋†
Kazumasa Yamamoto　　Shigeru Oyanagi

１．はじめに

開鍵暗号を初めて実現したのが RSA 暗号であり、RSA 暗号はデジタル署名が可能な方式でもある。そのため、RSA 暗号は公開鍵暗号として実際によく利用されている。RSA 暗号は解読できない安全な暗号強度を得るために、1024bit 以上の剰余乗算とそれを繰り返すべき乗演算を行う必要がある。そこで、大きな桁数を 演算するため桁上げ伝播を抑えることが出来、冗長二進数を用いて加算器、乗算器を構成し、剰余乗算とべき乗演算を高速で行うアルゴリズム、モンゴメリ乗 算、右向きアレイ法を用いて RSA 暗号回路を設計し実行時間を計測する。

2．RSA 暗号

RSA 暗号の処理は大きく分けて鍵作成、暗号化、復号化の三つに分けられる。鍵作成の流れは２つの大きな素数 p，q を選択し、n＝pq と φ(n)=(p-1)(q-1)を求める。 この n を法と呼ぶ。次に、φ(n)と e の最大公約数が 1 となる e を選択する。この e を公開指数と呼び、この公開指数と法が公開鍵となる。1=de mod φ(n)となる d を求める。この d を秘密指数と呼び、p、q、d が秘密鍵となる。
暗号化、復号化に用いるのは次の式である。
平文 m から暗号文 c を作成する

$c = m^e (\mathrm{mod}\ n)$

暗号文 c から元の平文 m を得る

$m = c^d (\mathrm{mod}\ n)$

3．冗長二進表現

冗長 2 進数表現とは、{-1、0、1}を用いて 2 進数を表現する方法である。

3.1　冗長二進表現を用いた加算

冗長二進を用いた加算は二つのステップで行い、x_n と y_n を足すとすると第 1 ステップでは各桁で $x_i + y_i = 2c_i + d_i$ を満たすように、中間桁上げ c_i+1 と中間和 d_i を求める。第 2 ステップで各桁の中間和 d_i と c_i を加え合わせ和を求める。

次に第 1 ステップを表 1、第 2 ステップを表 2 に示す。

表 1　第 1 ステップ

x_i ＼ y_i	-1	0	1
-1	-1,0	*0, -1/-1,1	*1,-1/0,1
0	*0,-1/-1,1	0,0	*1,-1/0,1
1	0,0	*1,-1/0,1	1,0

表 2　第 2 ステップ

d_i ＼ c_i	-1	0	1
-1	×	-1	0
0	-1	0	1
1	0	1	×

3.2　冗長二進表現を用いた乗算

部分積を冗長二進数とみなし、冗長二進加算を基本計算として二分木状に加え合わせ、最後に冗長二進表現から二進表現に変換する。

4．モンゴメリ乗算

剰余演算を高速に行うアルゴリズムとして、a*b(mod N)を求めるモンゴメリ乗算を説明する。まず、モンゴメリ乗算で扱う値を説明する。a と b を入力、法はNとする。モンゴメリ定数と呼ばれるRは 2 のべき乗であり、法Nより大きい数字の中で最も小さい値をとる。モンゴメリ乗算の特徴は剰余の性質を上手く利用することで、下位 r bit を 0 に調節することである。N で割った余りを求める場合、除数に N の整数倍を足しても答えが変わることはない。a*b + N*Q の値の下位 r bit を 0 となるような V の値はユークリッド互除法を行うことによって求めることができる。次に、演算手順を表 3 に示す。

表 3　モンゴメリ乗算の演算手順

Input	a, b
Output	$Y = a*b*R^{-1} \bmod N$
前計算	$V = -N^{-1} \bmod R$ $R = 2^r \quad (2^r > N > 2^{r-1})$
Step1	Q = a*b*V mod R
Step2	M = (a*b + N*Q) / R
Step3	if(M >= N)　Y = M – N else　Y = M

5．右向きアレイ法

べき指数を k bit で l 個に分割する場合、前処理演算として、m, m^2, m^3, ……, $m^{2\wedge k-1}$ の値を用意しておかなければならず、前計算に 2^k -2 回の演算を必要とする。

前計算を行った後、変数 X に入力値 m を e[l-1]回乗算した値を入力し、変数 X を k 回乗算する。この作業を分割した l 回分行うと m^e を求めることが出来る。この方法での計算回数は前計算に必要な 2^k -2 回と l を足した回数である。

以上のことをまとめて表 4 に示す。

表 4　右向きアレイ法

Input	m, e[l-1]
Output	m^e
前計算	$M[2^k] = \{ m^{2\wedge k-1}, m^{2\wedge k-2}, \ldots\ldots, m\}$
Step1	X = M[e[l-1]]
Step2	for (i = 0; i<=k; i=i+1) X = X*X
Step3	l = l – 1
Step4	X = X * M[e[l-1]]
Step5	if (l-1 != 0)　return to Step2

†立命館大学

6．構成方法

べき乗算を行うには右向きアレイ法のアルゴリズムに基づきモジュールを構成し、剰余演算を行うにはモンゴメリ乗算アルゴリズムに基づいてモジュールを構成する。べき乗演算は剰余乗算を繰り返し行うので、演算を行うのは剰余乗算を行うモジュールだけである。さらに剰余演算を行うモジュールは加算と乗算しか行わない。その加算と乗算は冗長二進表現を用いて構成する。冗長二進表現を用いた乗算器では部分積を求める場合に図 1 の様に二分木状に加え合わせる。最後に求めた積を桁上げ先見加算器で冗長二進表現を二進表現に変更する。

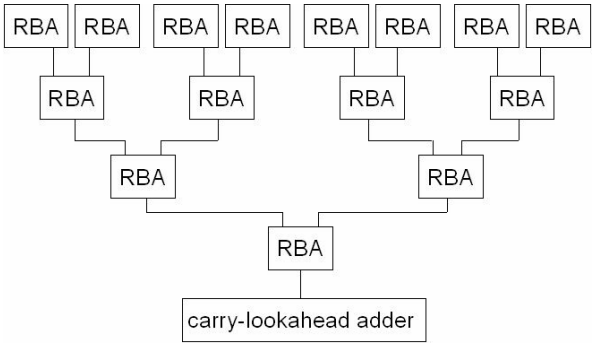

図 1　冗長二進乗算器構造図

以上の様な構成および冗長二進表現を用いなかった構成について、使用言語 Verilog HDL、開発ツールは Xilinx 社の Project Navigator 6.1i と ModelSim XE III 6.0d を用いて速度とゲート数を求めた。

7．実装結果

冗長二進表現を用いなかった場合

表 5　実行結果

N のビット数	1024
実行時間 (ms)	9.447
最高クロック周波数(MHz)	9.725
ゲート数	30645

冗長二進表現を用いた場合

表 6　実行結果

N のビット数	1024
実行時間 (ms)	4.450
最高クロック周波数(MHz)	20.642
ゲート数	50024

8．評価

冗長二進表現を用いた場合、用いない場合と比べて実行時間は 0.47 倍になり、最高クロック周波数は 2.12 倍になった。しかし、ゲート数は 1.63 倍に増加している。

9．検討

速度の向上は見られたもののゲート数が増加した。冗長二進表現を用いた場合一般的にはゲート数は倍に増えるが、桁上げ伝播がないことからモンゴメリ乗算で用いた多倍長演算回路が必要なくなることから今回は倍には増加しなかったと考えられる。

また、さらに速度の向上を求める場合は冗長二進表現の代わりに RSN 表現などを用いることが考えられる。

10．おわりに

本研究では RSA 暗号の剰余演算とべき乗演算にそれぞれモンゴメリ乗算と右向きアレイ法を用いて実装した。さらに、モンゴメリ乗算で行う加算と乗算に冗長二進加算と冗長二進乗算を用いた。結果、冗長二進表現を用いた場合は用いない場合と比べて実行時間が 0.47 倍となった。さらに、速度の向上を求める場合、冗長二進表現の代わりに RNS 表現を用いることが考えられる。

参考文献

[1] R. L. Rivest, A. Shamir, and L. Adliman: “A Method for Obtainning Digital
Signatures and Public-key Cryptosystems,” Communications of the ACM,
21(2), pp.120-126, Feb. 1978.

[2] P. L. Montgomery: “Moduler Multiplication Without Trial Division,” Mathematics
of Computation, 44(170), pp.519-521, Apr. 1985.

[3]吉原智明：高速モンゴメリ乗算回路に関する研究、北陸先端技術大学院大学修士論文　2003

[4]　松村暢也：冗長二進数を用いた RSA 用乗算器の高速化、高知工科大学修士論文　2003

[5]　高木直史：算術演算の VLSI アルゴリズム, コロナ社, 2005