Add-On Security Model for Public-Key Cryptosystem Based on Magic Square Implementation

Gopinanath Ganapathy, and K. Mani

***Abstract*—The efficiency of a cryptographic algorithm is based on its time taken for encryption / decryption and the way it produces different cipher text from a clear text. The RSA, the widely used public key algorithm and other public key algorithms may not guarantee that the cipher text is fully secured. As an alternative approach to handling ASCII characters in the cryptosystems, a magic square implementation is thought of in this work. It attempts to enhance the efficiency by providing add-on security to the cryptosystem. This approach will increase the security due to its complexity in encryption because it deals with the magic square formation with seed number, start number and sum that cannot be easily traced. Here, encryption / decryption is based on numerals generated by magic square rather than ASCII values. This proposed work provides another layer of security to any public key algorithms such as RSA, ElGamal etc., Since, this model is acting as a wrapper to a public key algorithm, it ensures that the security is enhanced. Further, this approach is experimented in a simulated environment with 2, 4, 8, and 16 processor model using Maui scheduler which is based on back filling philosophy.**

***Index Terms*—Magic Square, Public Key Cryptosystem, RSA, Security.**

I. Introduction

Cryptography is the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication, and data origin authentication. Cryptographic algorithms are divided into public-key and secret-key algorithms. In public-key algorithms both public and private keys are used, with the private key computed from the public key. Secret-key algorithms rely on secure distribution and management of the session key, which is used for encrypting and decrypting all messages. Though, public-key encryption is slower than symmetric-key encryption, it is used for bulk-data encryption. This is also due to encryption/decryption processes. Hence, in practice cryptosystems are a mixture of both [1], [12].

There are two basic approaches used to speed up the cryptographic transformations. The first approach is to design faster (symmetric or asymmetric) cryptographic algorithms. This approach is not available most of the time. The speed of cryptographic algorithm is typically determined by the number of rounds (in private-key) or by the size of messages (in public-key case). The second approach is the parallel cryptographic system. The main idea is to take a large message block, divide it into blocks of equal sizes and each block can be assigned to one processor [7]. To perform the operations in parallel, effective scheduling is very important so that it can reduce the waiting time of message for processing. The back filling is one such approach.

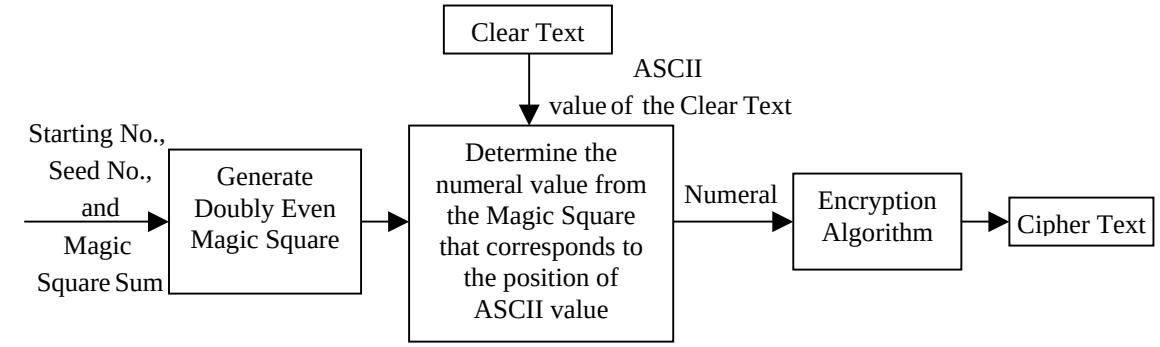
The security of many cryptographic systems depends upon the generation of unpredictable components such as the key stream in the one-time pad, the secret key in the DES algorithms, the prime p, and q in the RSA encryption etc. In all these cases, the quantities generated must be sufficient in size and the random in the sense that the probability of any particular value being selected must be sufficiently small. However, RSA is not semantically secure or secure against chosen cipher text attacks even if all parameters are chosen in such a way that it is infeasible to compute the secret key d from the public key (n, e), choosing p, q are very large etc. Even if the above said parameters are taken carefully, none of the computational problems are fully secured enough [8]. Because to encrypt the plaintext characters, their ASCII values are taken and if a character occurs in several places in a plaintext there is a possibility of same the cipher text is produced. To overcome the problem, this paper attempts to develop a method *with different doubly even magic squares of order 16* and each magic square is considered as one ASCII table. Thus, instead of taking ASCII values for the characters to encrypt, preferably different numerals representing the position of ASCII values are taken from magic square and encryption is performed using RSA cryptosystem.

II. Methodology

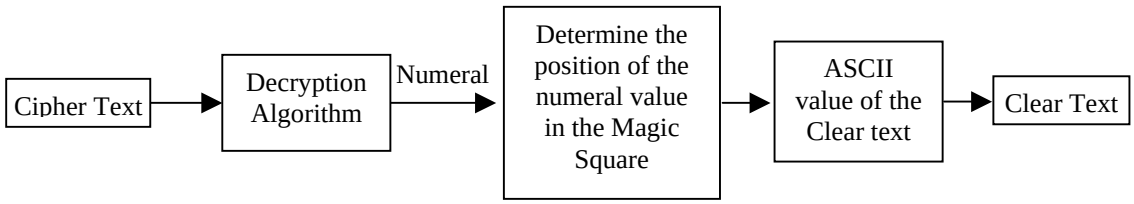
The working methodology of the proposed add-on security model is discussed stepwise.

- Construct different doubly even magic square of order 16 as far as possible and each magic square corresponds to one ASCII set.
- To encrypt the character, use the ASCII value of the character to determine the *numeral in the magic square* by considering the position in it. Let N_P and N_C denote the numeral of the plaintext and cipher text respectively. Based on N_P and N_C values, all plaintext and cipher text characters are encrypted and decrypted respectively using RSA algorithm.

Manuscript received July 15, 2009.
Dr. Gopinath Ganapathy, Professor and Head with the computer Science Department, University of Bharthidasan, Trichy, India – 620 024. (Phone : +91 9842407008; email: gganapathy@gmail.com).
K. Mani is with the computer Science Department of Nehru Memorial College, Puthanapatti, University of Bharthidasan, Trichy, India – 621 007. (Phone : +91 9443598804; email: nitishmanik@yahoo.com).



(a) Encryption process



(b) Decryption process

Fig. 1. Add-On Wrapper Model

- To speed up the operations, perform them in parallel in a simulated environment. For that, use Maui scheduler with back filling philosophy.

The methodology of Add-on Security Model is shown in Fig. 1.

A. Magic Squares and their construction

Definition 1: A magic square of order n is an arrangement of integers in an n*n matrix such that the sums of all the elements in every row, column and along the two main diagonals are equal.

A normal magic square contains the integers from 1 to n^2. It exists for all orders $n \geq 1$ except n = 2, although the case n = 1 is trivial as it consists of a single cell containing the number. The constant sum in every row, column and diagonal is called the magic constant or magic sum, M [6]. The magic constant of a normal magic square depends only on n and has the value

$$M(n) = \frac{n(n^2 + 1)}{2} \quad (1)$$

For normal magic squares of order n = 3, 4, 5, 6, 7, 8 … the magic constants are 15, 34, 65, 111, 175, 265… respectively. Though there is only one magic square of order 3 apart from trivial notations and reflections, the number of squares per order quickly sky rockets. There are 880 distinct order 4 squares, and 275,305,234 distinct order 5 magic squares [2], [4]. Magic squares can be classified into three types: odd, doubly even (n divisible by four) and singly even [3] (n even but not divisible by four). This paper focuses only on doubly even magic square implementation and their usefulness for public-key cryptosystem

A.1. Construction of Doubly Even Magic Square

In this work, to generate the doubly even magic square, any seed number, starting number, and magic sum may be used and the numbers generated will not be in consecutive order.

The algorithm (*A.2*) starts with building 4 x 4 magic square. Incrementally 8 x 8 and 16 x 16 magic squares are built using 4 x 4 magic squares as building blocks. While constructing the doubly even magic squares the following *notations* are used.

MS : *Magic Square*
n : *Order of MS where n = 4m, where m = 1, 2, 3 and 4*
MSn : *MS of order n*
MSB4 : *Base MS of order 4*
MS_{start} : *Starting number of MS*
$MSTn_{sum}$: *Total sum of MS of order n*
$MSDn_{sum}$: *Diagonal sum of MS of order n*
T_No_MS : *Total Number of MS*
$S_1 S_2 S_3 S_4$: *First, Second, Third and Fourth digit of 4 digits seed number. Each digit has a value from 0 to 7. If $S_1 = S_2 = S_3 = S_4 = 0$, i.e 0000 then it is MSB4. The values in the MSB4 are filled as shown in Fig. 2. Call it as MS4_fill_order (Min $_{start}$, Max $_{start}$)*

-4	MS_{start}	-8	+12
-10	+14	-6	+2
+8	-12	+4	$MST4_{sum}$
+6	-2	+10	-14

Fig. 2. Magic Square Filling Order

where –int represents the places to fill the values in MS, starting from $MST4_{sum}$ and decremented by 2 each time to get the next number and +int represents the places to fill the values in MS, starting from MS_{start} and incremented by 2 to get the next number.

MS4_min_start_value : *Minimum starting value of MS*
MS4_fill_order (MS_{start}, $MST4_{sum}$) : *function used to fill the values in MS in the order shown in Fig. 2*
MSn_base i : *ith n^{th} order base MS*

MS4_sub i : *ith 4^{th} order MS obtained from MSB4 by using suitable transformation based on S_i, where i= 1,2,3,4*

i, j, count : *Index variables for loop*

|| : *Concatenation*

A.2. Algorithm

Input : *4 digit Seed number, Starting Number and Magic Square Sum*

Output: *Doubly Even Magic Square of order 16.*

1. ***Read*** MS_{start} ; S_i, i = 1,2,3,4 ;$MST16_{sum}$ and T_No_MS
2. $MST4_{sum} \leftarrow \lceil MST16_{sum} / 8 \rceil$
3. $MST8_{sum} \leftarrow \lceil MST16_{sum} / 4 \rceil$
4. Count $\leftarrow 1$
5. ***While*** Count $\leq$ T_No_MS do // to generate MS16

5.a. prev$\leftarrow$ 1

5. b. ***for*** j $\leftarrow$ 1 to 4 // to generate four MS8

5.b.1 ***for*** i $\leftarrow$ 1 to 4 // to generate four MS4

5.b.1.1 ***if*** ((j=2)&&(i = 3)) || ((j = 3) &&(i =2))

MS4_min_start_value $\leftarrow$ $MS4_{start}$ + 16 *(iprev-1) +1

else

MS4_min_start_value $\leftarrow$ MS_{start} + 16* (prev-1)

end if

5.b.1.2 MS4_max_start_value $\leftarrow$ $MST4_{sum}$ – 16* (prev- 1) - MS_{start}

5.b.1.3 MS4_base i $\leftarrow$ call MS4_fill_order (MS4_min_start_value, MS4_max_ start_value)

5.b.1.4 ***Case*** S_i in 0,1,2,3,4,5,6,7

0: MS4_sub i $\leftarrow$ MS4_base i

1: MS4_sub i $\leftarrow$ rotate MS4 _base i by 90°

2: MS4_sub i $\leftarrow$ rotate MS4_base i by 180°

3: MS4_sub i $\leftarrow$ rotate MS4_base i by 270°

4: MS4_sub i $\leftarrow$ rotate MS4_base i by 180° along vertical axis

5: MS4_sub i $\leftarrow$ rotate MS4_base i by 90° through anticlock wise

6: MS4_sub i $\leftarrow$ rotate MS4_base i by 180° along horizontal axis

7: MS4_sub i $\leftarrow$ rotate MS4_base i by 180° diagonally

end case S_i

5.b.1.5 prev$\leftarrow$ prev+1

end for i

5.b.2 MS8_sub j $\leftarrow$MS4_sub 1|| MS4_sub 2 || MS4_sub 3 || MS4_sub 4

5.b.3 ***case*** j in 2,3,4

2: $S_1 \leftarrow S_2$; $S_2 \leftarrow S_3$; $S_3 \leftarrow S_1$; $S_4 \leftarrow S_4$

3: $S_1 \leftarrow S_3$; $S_2 \leftarrow S_1$; $S_3 \leftarrow S_4$; $S_4 \leftarrow S_2$

4: $S_1 \leftarrow S_4$; $S_2 \leftarrow S_1$; $S_3 \leftarrow S_3$; $S_4 \leftarrow S_2$

end case j

5.b.4 $S_i \leftarrow S_1 || S_2 || S_3 || S_4$, i $\leftarrow$ 1,2,3,4

end for j

5.c MS16_count $\leftarrow$ MS8_sub1|| MS8_sub2 ||MS8_sub3||MS8_sub4

5.d $S_i \leftarrow (S_i + rnd(10)) \bmod 8$, i = 1,2,3,4

5.e count $\leftarrow$ count + 1

end while count

A.3. Magic Square Construction - Example

1. Let $MS_{start} = 4$; $S_1 = 0$, $S_2 = 1$, $S_3 = 0$, $S_4 = 0$; $MST16_{sum} = 12345$ and T_No_MS =4
2. $MST4_{sum} = \lceil 12345/8 \rceil = \lceil 1543.13 \rceil = 1543$
3. $MST8_{sum} = \lceil 12345/4 \rceil = \lceil 3086.25 \rceil = 3086$
4. Count =1
5. While count $\leq$ 4

5.a prev=1

5.b j = 1

5.b.1 i = 1

5.b.1.1 MS4_min _start _value= 4+16(0)= 4

5.b.1.2 MS4_max _start _value = 1543–16(0)–4=1539

5.b.1.3 MS4_base 1 = call fill_order(4,1539)

1535	4	1531	16
1529	18	1533	6
12	1527	8	1539
10	1537	14	1525

Fig. 3. MS4_sub1 with (4, 1539)

5.b.1.4 since $S_1 = 0$ MS4_sub 1 = MS4_base 1

5.b.1.5 prev=2

5.b.1 i = 2

5.b.1.1 MS4_min _start _value = 4+ 16(1)=20

5.b.1.2 MS4_max _start _value =1543–16(1)–4=1523

5.b.1.3 MS4_base 2 = call fill_order (20,1523)

1519	20	1515	32
1513	34	1517	22
28	1511	24	1523
26	1521	30	1509

Fig. (a). MS4_base 2 with (20, 1523)

5.b.1.4 since $S_2 = 1$

MS4_sub 2 = 90° rotation of MS4_base 2

26	28	1513	1519
1521	1511	34	20
30	24	1517	1515
1509	1523	22	32

Fig. (b). 90° rotation of Fig. (a)

Fig. 4. MS4_sub2 with (20, 1523)

5.b.1.5 prev= 3

5.b .1 i = 3

5.b.1.1 MS4_min _start _value = 4+ 16(2) = 36

5.b.1.2 MS4_max _start _value =1543–16(2)–4 =1507

5.b.1.3 MS4_base 3 = call fill_order (36,1507)

1503	36	1499	48
1417	50	1501	38
44	1495	40	1507
42	1505	46	1493

Fig. 5. MS4_sub3 with (36, 1507)

5.b.1.4 since $S_3 = 0$ MS4_sub 3 = MS4_base 3

5.b.1.5 prev = 4

5.b.1 i = 4

5.b.1.1 MS4_min _start _value = 4+ 16(3) = 52

5.b.1.2 MS4_max _start _value =1543–16(3)–4 = 1491

5.b.1.3 MS4_base 4 = call fill_order(52,1491)

1487	52	1483	64
1481	66	1485	54
60	1479	56	1491
58	1489	62	1477

Fig. 6. MS4_sub4 with (52, 1491)

5.b.1.4 since $S_4 = 0$ MS4_sub 4 = MS4_base 4

5.b.2 MS8_sub 1 = MS4_sub1 || MS4_sub2 || MS4_sub3 || MS4_sub4

Other matrices MS8_sub2, MS8_sub3 and MS4_sub4 are generated in this manner and they are concatenated so that they form MS16 which is shown in Fig. 7. Similarly other MS16 matrices are generated by using suitable transformations of the seed number. In this paper, only four MS16 matrices are generated.

B. Encryption / Decryption of message using RSA cryptosystem with Magic Square

To show the relevance of this work to the security of public-key encryption schemes, a public-key cryptosystem RSA is taken. RSA was proposed by Rivest et al [10], [11]. The private key of a user consists of two prime p and q and an exponent (decryption key) d.

The public-key consists of the modulus n = pq, and an exponent e such that $d = e^{-1} \bmod (p-1)(q-1)$. To encrypt a plaintext M the user computes $C = M^e \bmod n$ and decryption is done by calculating $M = C^d \bmod n$. In order to thwart currently known attacks, the modulus n and thus M and C should have a length of 512-1024 bits in this paper.

B.1. Wrapper implementation – Example

In order to get a proper understanding of the subject matter of this paper, let p = 11, q = 17 and e = 7, then n = 11(17) = 187, (p-1)(q-1) = 10(16) = 160. Now d = 23. To encrypt, $C = M^7 \bmod 187$ and to decrypt $M = C^{23} \bmod 187$. Suppose the message is to be encrypted is "BABA". The ASCII values for A and B are 65 and 66 respectively. To encrypt B, the numerals which occur at 66^{th} position in first (Fig.7) and third MS16(not shown here) are taken because B occurs in first and third position in the clear text. Similarly, to encrypt A, the numerals at 65^{th} position in second and fourth MS16(not shown here) are taken. Thus $N_P(A) = 42$ and 48, $N_P(B) = 36$ and 44. Hence $C(B) = 36^7 \bmod 187 = 9$, $C(A) = 42^7 \bmod 187 = 15$, $C(B) = 44^7 \bmod 187 = 22$, $C(A) = 48^7 \bmod 187 = 157$. Thus, for same A and B which occur more than once, different cipher texts are produced for the same.

C. Parallel Cryptography

To speed up cryptographic transformation, the parallel cryptography is used. Effective scheduling is important to improve the performance of crypto system in parallel [13]. The scheduling algorithms are divided into two classes: time sharing and space sharing. Backfilling is the space sharing optimization technique.

Maui is a job scheduler specifically designed to optimize system utilization in policy driver, heterogeneous high performance cluster (HPC) environment. The philosophy behind it is essentially schedule jobs in priority order, and then backfill in the holes [9]. Maui has a two-phase scheduling algorithm. In the first phase, the high priority jobs are scheduled using advanced reservation. A backfill algorithm is used in the second phase to schedule low-priority jobs between previously selected jobs [5].

In our study the Maui Scheduler with back filling scheduling technique is used to calculate the encryption/decryption time of given message in simulated environment.

35	4	1531	16	26	28	1513	1519	74	76	1465	1471	1455	85	1451	96
1529	18	1533	6	1521	1511	34	20	1473	1463	82	68	1449	98	1454	86
12	1527	8	1539	30	24	1517	1515	78	72	1469	1467	93	1447	88	1459
10	1537	14	1525	1509	1523	22	32	1461	1475	70	80	90	1457	94	1446
1503	36	1499	48	1487	52	1483	64	1439	101	1435	112	1423	116	1419	128
1497	50	1501	38	1481	66	1485	54	1433	114	1438	102	1417	130	1421	118
44	1495	40	1507	60	1479	56	1491	109	1431	104	1443	124	1415	120	1427
42	1505	46	1493	58	1489	62	1477	106	1441	110	1430	122	1425	126	1413
1407	132	1403	144	1391	149	1387	160	1343	196	1339	208	1327	212	1323	224
1401	146	1405	134	1385	162	1390	150	1337	210	1341	198	1321	226	1325	214
140	1399	136	1411	157	1383	152	1395	204	1335	200	1347	220	1319	216	1331
138	1409	142	1397	154	1393	158	1382	202	1345	206	1333	218	1329	222	1317
1375	165	1371	176	186	188	1353	1359	1311	228	1307	240	250	252	1289	1295
1369	178	1374	166	1361	1351	194	180	1305	242	1309	230	1297	1287	258	244
173	1367	168	1379	190	184	1357	1355	236	1303	232	1315	254	248	1293	1291
170	1377	174	1366	1349	1363	182	192	234	1313	238	1301	1285	1299	246	256

Fig. 7. First MS16 using Starting No. 4, Seed No. 0100, and Magic Sum 12345

File Size (MB)	No. of Processors														
	1			2			4			8			16		
	E ms	D ms	T ms	E ms	D ms	T ms	E ms	D ms	T ms	E ms	D ms	T ms	E ms	D ms	T ms
1	328	361	689	518	540	1058	440	462	902	401	420	821	381	392	773
2	640	641	1281	682	690	1372	518	528	1046	440	455	895	402	404	806
4	1250	1297	2547	994	1010	2004	678	682	1360	520	522	1042	441	443	884
8	2515	2578	5093	1627	1651	3278	994	1006	2000	680	686	1366	520	524	1044

E- Encryption time, D-Decryption time, T- Total Time, and ms- milliseconds

Fig. 8. Encryption and Decryption time using RSA on a Pentium Processor

III. Experimental result

The methodology proposed is implemented in visual C++ version 6.0. The time taken for encryption and decryption of various file sized message in simulated parallel environment using RSA public key crypto system with magic square are shown in Fig. 8. The simulation scenario configured in our implementation consisting of 2, 4, 8, and 16 processors.

From Fig. 8, we observe that as the file size is increased in double, encryption and decryption time is also increased in double for single processor. Moreover, the time taken for encryption and decryption is almost same. Parallel encryption and decryption have more effect if the file size is increased.

IV. Conclusion

An alternative approach to existing ASCII based cryptosystem a number based approach is thought of and implemented. This methodology will add-on one more layer of security, it adds numerals for the text even before feeding into public-key algorithm for encryption. This add-on security is built using magic square position equivalent to the character to be encrypted.

Further efficiency of cryptographic operation depends on performing them in parallel. Simulation using different number of processors for encryption /decryption has shown that the time taken for decryption is approximately 1.2 times larger than the corresponding time for encryption.

References

[1] Antti Hamalainenn, Matti Tommiska, and Jorma Skytta, "*6.78 Gigabits per Second Implementation of the IDEA Cryptographic Algorithm LNCS 2438*", Springer-Verlag, 2002, pp. 760-769.

[2] Adam Rogers, and Peter Loly, "*The Inertial Properties of Magic Squares and Cubes"*, Nov. 2004, pp. 1-3.

[3] Clifford A. Pickover, "*The Zen of Magic Squares, Circles and Stars"*, Universities Press, Hyderabad, India, 2002.

[4] Dan Thomasson. *Knights Templar Ciphers. March 2004*, Available: http://www.borderschess.org/ KTcipher.html

[5] Daniel Page, and Nigel P.Smart. "*Parallel Cryptographic Arithmetic using a redundant Montegomery Representation"*. IEEE Transactions on Computers Vol. 53, No.9, November 2004.

[6] http://en.wikipedia.org/wiki/Magic_squares , pp. 1-3.

[7] Josef Piepryk, and David Pointcheval, "*Parallel Authentication and Public-Key Encryption"*, The Eigth Australasian Conference on Information Security and Privacy, Springer-Verlag, Jul. 2003, pp. 383-401.

[8] A. J. Menezes, P.C. Van Oorschot, and S. Vanstone, "*Handbook of Applied Cryptography"*, CRC Press, Boca Ration, Florida, USA, 1997.

[9] Sayeed Iqbal, Rinku Gupta and Yung-Chin Lang, "*Job Scheduling in HPC Clusters"*, Power Solutions, Feb. 2005, pp. 133-135.

[10] B.Schneier, "*Applied Cryptography"*, John Wiley & Sons Inc., New York, Second Edition, 1996.

[11] Stallings,"*Cryptography and Network Security"*, Prentice Hall, Upper Saddle River, New Jersey, USA, Second Edition, 1997.

[12] Thomas Wollinger, Jorge Guajardo, and Chirst of Paar, "*Cryptography in Embedded Systems: An overview. In Proc. of the Embedded World 2003 Exhibition and Conference"*, Feb. 18-20, 2003, pp. 735-744,.

[13] Y.Zhang, H. Franke, J. E. Moreira and A. Sivasubramanian, "*An Integrated Approach to Parallel Scheduling using Gang-Scheduling, Backfilling and Migration"*, JSSPP, UK, Springer-Verlag, 2006, pp. 133-158.