Proposal for Portable Approach in Advance Encryption Standard

Ankit Sharma
Department of Computer Science & Engineering,
Lovely Professional University, Jalandhar, Punjab

Shashank Gupta
Department of Computer Science & Engineering,
Lovely Professional University, Jalandhar, Punjab

Shashi Kant Rathore
Department of Computer Science & Engineering,
Lovely Professional University, Jalandhar, Punjab

ABSTRACT
Cryptography is dealing with a lot of different algorithms which are much secure in various aspects but there are two major problems coming in the cryptographic algorithms, first the portability of algorithm from heavy applications to light applications and second the current Method of Formal Coding-Side Channel Attack (MFCSCA) which are targeting XOR function of the algorithms. To resolve these two problems we propose a new algorithm by using AES algorithm with lattice concept of multidimensionality. In this paper, we propose a new algorithm by combining the concepts of mathematics and multidimensionality concept of physics which solve both the problems of the encryption algorithms.

Keywords
Lattice based transpose, Feistel structure, S-Box, Key Permutation, Kirchhoff's principle, MFCSCA.

1. INTRODUCTION
As ubiquitous computing becomes a reality, sensitive information is increasingly processed and transmitted by smart cards, mobile devices and various types of embedded systems. This has led to the requirement of a new class of lightweight cryptographic algorithm to ensure security in these resource constrained environments. In January 2012, the International Organization for Standardization (ISO) has standardized two low-cost block ciphers for this purpose ,CLEFFIA and PRESENT [1], but the security of these algorithms are restricted to small applications only but for large applications again Advanced Encryption Standard (AES) and Data Encryption Standard (DES) are preferred.

The National Institute of Standards and Technology (NIST) selected the RJINDAEL algorithm as the new Advanced Encryption Standard (AES) in 2001. Numerous FPGA and ASIC implementations of the AES were previously proposed and evaluated. To date, most implementations feature high speeds and high costs suitable for high-end applications only. The need for secure electronic data exchange will become increasingly more important. Therefore, the AES must be extended to low-end customer products, such as PDAs, wireless devices, and many other embedded applications. In order to achieve this goal, the AES implementations must become very inexpensive [2] but so far slow.

In this paper, we propose method which is light weight and secure; and can be used by light as well as heavy applications this is achieved by combining the concepts of mathematics and physics which provide complex but fast method of encryption. The basic idea is to remove the XOR function from the algorithm which make the algorithm fast and because of the lattice based transposition and RJINDAEL algorithm [8] for substitution security is maintained. We focused on the various factors to maintain the secrecy of message and at what level security are needed, i.e. light application or heavy application, so on the basis of that factors key length is decided and algorithm will dynamically change its few factors which maintain the security. The minimum key length would be 128 bits which is much secure and does not easily captured by intruder. The reason behind removing XOR function is it decreases the implementation speed and recently intruders using Method of Formal Coding-Side Channel Attack(MFCSCA) to resolve XOR-sum [3][10][22], i.e. algebraic normal form (ANF), which result in XOR as an unsecure mode of operation for encryption.

Our method resolves the Method of Formal Coding-Side Channel Attack (MFCSCA) and without compromising the security and complexity of the encryption yet providing the much faster implementation than the AES and DES.

2. PRAPOSED PORTABLE ENCRYPTION ALGORITHM
As Feistel structure is one of the most widely used and best studied structures for the design of block ciphers. It was proposed by H. Feistel in the early 1970s; subsequently the structure was adopted in the well-known block cipher DES. During the 30-year of modern block cipher research history, extensive studies have been made on Feistel structure. Currently, many well-known Block ciphers employ the design of Feistel structures. On the other hand, an optimal diffusion which is a linear function with the maximum branch number is widely regarded in the recent block cipher research; the concept is used in the design of AES/RJINDAEL and many other cryptographic primitives[4][5][6]. Therefore, we also prefer Feistel structure in our designing concept. There are two inputs for the algorithm one is plain text to encrypt and secret key which size would be dynamic and change from one implementation to another but it will static for a particular implementation, the size of key would decide number of rounds and many other security factors of the algorithm, which would be discussed in next section. So our algorithm can be explained under following sections:

A. *Key size and key generation*
B. *Basic structure and organization of operations*
C. *Decryption*.

2.1 Key Size and Key Generation
The key size would be greater than or equal to 128 bits (keylen ≥ 128 bits). The initial key sequence would decide various factors for the algorithm:

a. The number of columns of the matrix of the message block would be ⌈keylen/8⌉.
b. Number of rounds would be ⌈keylen/4⌉.
c. The key sequence would decide the shift pattern which is used to generate keys for all the rounds.
d. The key sequence of every round would decide the column indexing of the matrix of the message by using hash indexing technique.

2.1.1 Key Generation

The key for every round would be generated by the left shifting of the previous round key, the shift would be decided by initial pattern of the key (figure 2), the basic structure for key generation is given in (figure 1:

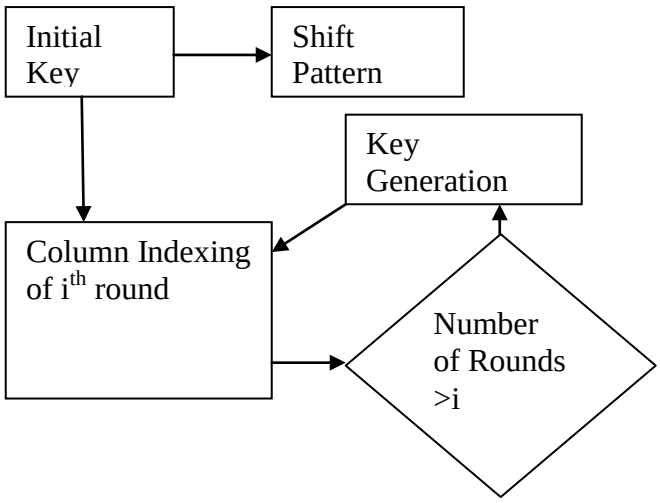

Fig 1. Basic structure of key generation

2.1.2 Shift pattern.

The shift pattern would be done by converting the binary numbers into decimal numbers, taking 4 bits for one decimal number, but due to the dynamic nature of the key sometimes it required padding, so shift pattern is done in two steps (figure 2):

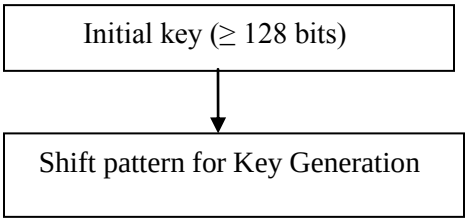

Fig 2. Finding the shift pattern for Key Generation

1. **Padding (optional):-** Adding zeros (0) at the end of the key to make it even, shown in table 1.

Key length (bits)	Number of bits
128	0 bits
129	3 bits (000)
130	2 bits (00)
131	1 bit (0)
132	0 bits

2. **Decimal conversion: -** Taking 4 bits for a decimal number.

Table 2. Showing decimal numbers for various bit sequence

Bit sequence	Equivalent decimal number
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	4
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

The key for the next round is generated by the circular left shift of the key of the previous round (figure 3).

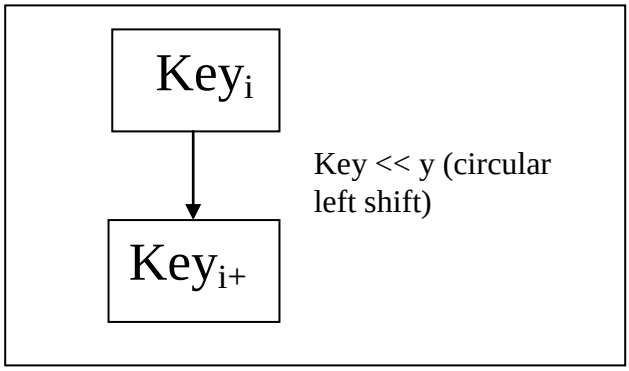

Fig 3. Key generation for the next round

The "y" shift is decided by the shift pattern which is generated by the initial key given by the user (shown in figure 2).

2.2 Basic Structure of Algorithm

The basic structure of algorithm is like the structure of AES, Advanced Encryption Standard; it contains four operations which are repeated in each round (figure 4).

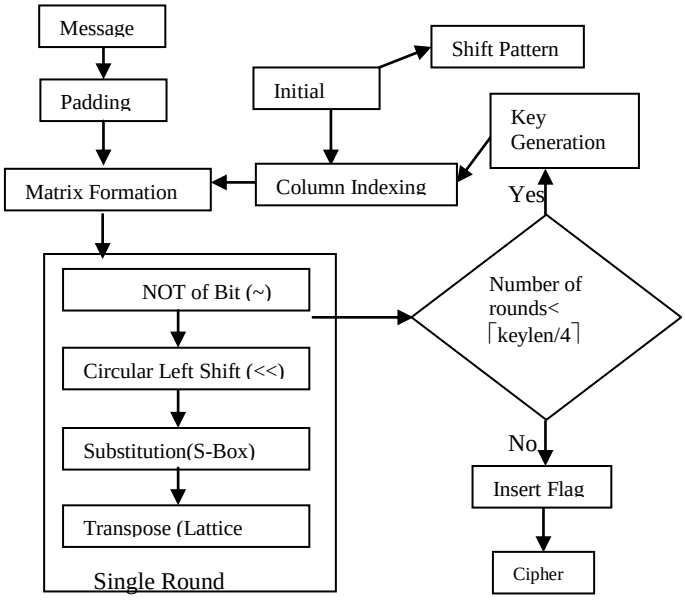

Fig 4. Basic Structure of the algorithm

a) Matrix formation and message padding

Matrix formation is also dynamic in nature it depends on the key length as explained in section A, the number of columns in the matrix would be $\lceil keylen/8 \rceil$, and in each column there would be 8 bits, i.e. 2 bytes, of the message which is filled row wise in the matrix. For example, let us take a keylen=128 bits , so $\lceil 128/8 \rceil$=16 , therefore the number of columns in the matrix would be 16 so the message is filled in the matrix as:

1	2	3							16
0C	12	97							DD

Fig 5. Sample Matrix for 128 bit key

Padding*:* As the message length is dynamic because of the portability of the algorithm to the light weight and heavy weight application, hence the block size of the algorithm is to be maintained, if the message size is; *n*(8* number of columns***)** where n=1, 2, 3……., i.e. $n \geq 1$,then the message is filled in the matrix row wise where n is the number of rows, else message padding is required. We choose the message padding technique used in SHA, Secure Hash Algorithm, [7]The purpose of this padding in SHA is to ensure that the padded message is a multiple of 512 or 1024 bits, depending on the algorithm. Padding can be inserted before hash computation begins on a message, or at any other time during the hash computation prior to processing the block(s) that will contain the padding. In the same way our purpose of the padding is to make the message block in the multiple of key length so if the message length is less than *(n*keylen)* we add padding bits to make it even, the padding consists of single 1 bit followed by necessary number of 0 bits.

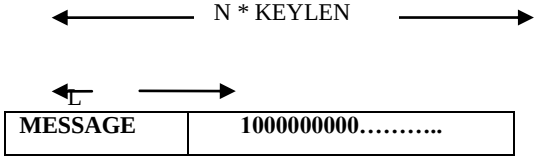

Fig 6. Message with padding

Column Indexing. After the pad is inserted to the message the message will be inserted into the matrix row wise, 8 bits in each column, and the indexing of the column would be done by finding *decimal sequence* of the key of that round, (table 2), as we done for finding the shift pattern for the key generation.

$I_i = D_i$

Where,

I_i=Index for i^{th} column.

D_i=Decimal Equivalent of the key for i^{th} column.

b) Our method has four modes of operations carrying out in every round:

1. *Not Of Bit(~):* As every column of the matrix has 8 bits so our algorithm flip the bit at :

   I_i *mod 8,* where I_i is the index of the i^{th} column.

2. *Circular left shift(<<):* Now in this step the circular left shit is done on the bits of the whole row by x times

   <<Xi

   Where, X_i=value (column with index of largest prime number in the i^{th} round key).

3. *S-Box Substitution:* In the next step we substitute the bits with the help of RJINDAEL S-Box shown in figure 7.

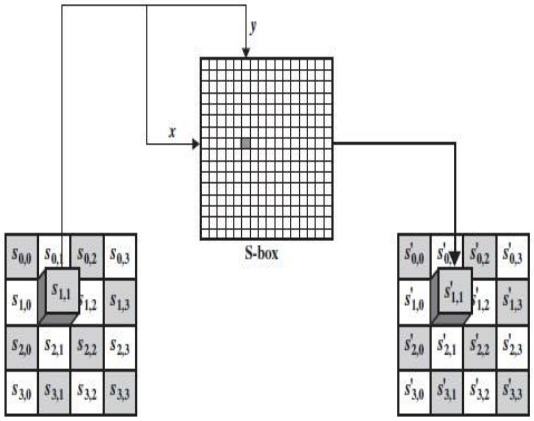

Fig 7. Substitute byte Transformation [8]

There are different S-Boxes for encryption (figure 8(a)) and decryption (figure 8(b)).

x \ y	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Fig 8(a). S-Box[8]

x \ y	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
C	1F	DD	A8	33	88	07	C7	31	B1	]12	10	59	27	80	EC	5F
D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

Fig 8(b). Inverse S-Box [8]

4. *Lattice Transpose:* This is the last operation of a round in this transposition of the bits is done but instead of using any

matrix or array algorithm uses the multidimensional lattice structure[20] (figure 9), transpose is done as:

1) Divide the message bits into ⌈Message Length/Xi⌉ parts, Where, X_i=value (column with index of largest prime number in the i^{th} round key). i.e., each part contains X_i bits.

2) Now apply circular left shift to each part separately to X_i times ($<<X_i$).

3) Then recombine all the parts and apply a circular right shift to whole message to X_i times ($>>X_i$).

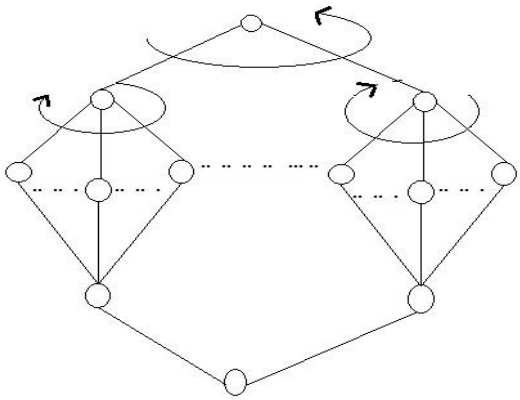

Fig 9. Multidimensional structure for transposition

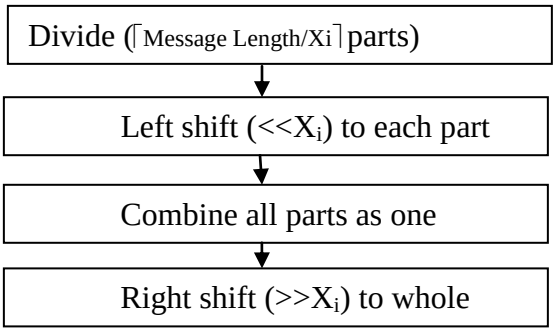

Fig 10. Steps Showing Lattice Based Transposition

c) Insert Flag*:* It is the extra bit which is added to the cipher after all the rounds are completed, flag shows whether the padding is done in the message or not, If padding is done in the message then flag would be "1" else "0".

It is added at the beginning of the cipher.

2.3 Decryption

It is just the reverse process of the encryption (figure 11) but the key generation section is changed (section a)

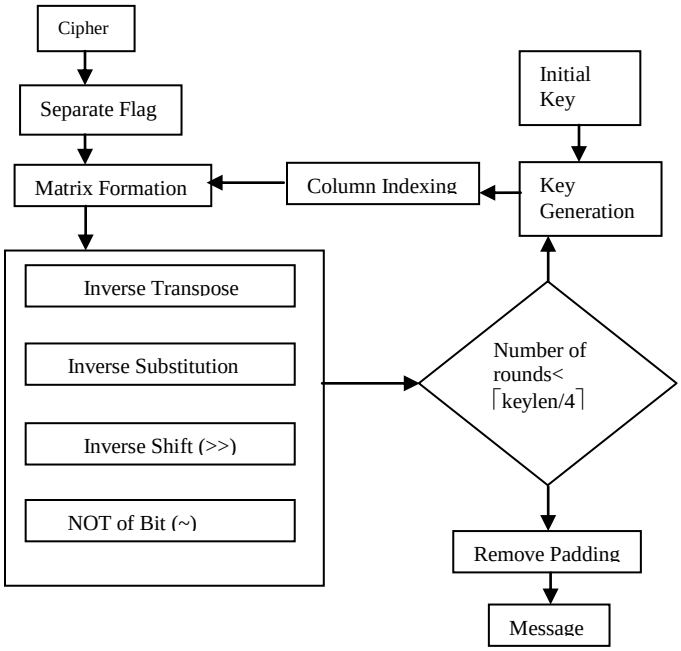

Fig 11. Structure for Decryption

a) **Key Generation***:* Key generation of decryption is same as encryption (section 2.1) with some changes in shifting as:
1. The first round key would be (<<Y), where Y is the sum of all the values of the shift pattern array.
2. Key for next rounds would be (>>Zj) , where Zj is the value at the j position of the shift pattern array and j would be ⌈keylen/8⌉,-1,….1.

3. CONCLUSION

Our proposed algorithm is resolving Method of Formal Coding-Side Channel Attack (MFCSCA) as well as the problem of portability of the algorithm to different applications. According to Kerckhoff [9], key is the only thing which is needed to be kept hidden hence we make key length a great importance and its length would be a great preference and though our algorithm not using XOR in any of its operation hence time complexity and problem of Method of Formal Coding-Side Channel Attack (MFCSCA) is resolved.

This algorithm is very useful for distributed networks and mobile applications as both of them have light weight applications but require high security. So, it is would be very useful if ASPE implemented on distributed environment like cloud, grid or mobile applications like android.

4. REFERENCES

[1] Changyong Peng , Chuangying Zhu , Yuefei Zhu ,Fei Kang; ”Symbolic computation in block cipher with application to PRESENT”;

[2] Pawe_l Chodowiec and Kris Gaj;” Very Compact FPGA Implementation of the AES Algorithm”.

[3] Changyong Peng, Chuangying Zhu, Yuefei Zhu, Fei Kang ;” Improved side channel attack on the block cipher NOEKEON”.

[4] P. S. L. M. Barreto and V. Rijmen, "The Whirlpool hashing function.";Primitive submitted to NESSIE, Sept. 2000.

[5] J. Daemen and V. Rijmen, “The Design of Rijndael: AES - The Advanced Encryption Standard”; (Information Security and Cryptography). Springer, 2002.

[6] Taizo Shirai and Kyoji Shibutani” ;On Feistel Structures Using a Di_usion Switching Mechanism”.

[7] FIPSPUB 180-4, FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION: Secure Hash Standard (SHS); Information Technology Laboratory National Institute of Standards and Technology Gaithersburg, MD 20899-8900,March 2012

[8] William Stallings;” Cryptography and Network Security Principles and Practice, 5th Edition”; Copyright © 2011, 2006 Pearson Education, Inc., publishing as Prentice Hall.

[9] J. Daemen and V. Rijmen. AES proposal: Rijndael, September 2001. http://csrc.nist.gov/CryptoToolkit/aes/rijndael/Rijndael.pdf.

[10] John Kelsey, Bruce Schneier, and David Wagner, “Related-Key Cryptanalysis of 3-WAY, Biham- DES, CAST, DES-X, NewDES, RC2, and TEA”. In *Information and Communications Security—Proceedings of ICICS 1997*, Lecture Notes in Computer Science 1334, Springer-Verlag, 1997.

[11] Yang, L., Wang, M., Qiao, S.: Side Channel Cube Attack on PRESENT. In: Garay, J.A., Miyaji, A., Otsuka, A. (Eds.) CANS 2009. LNCS, vol. 5888, pp. 379-391. Springer, Heidelberg (2009).

[12] Sumio Morioka and Akashi Satoh. A 10 Gbps full-AES crypto design with a twisted-BDD S-box architecture. In *IEEE International Conference on Computer Design*. IEEE, 2002.

[13] Vincent Rijmen. Efficient implementation of the RJINDAELS-box. available at http://www.esat.kuleuven.ac.be/~rijmen/rijndael/sbox.pdf, 2001.

[14] Atri Rudra, Pradeep K. Dubey, Charanjit S. Jutla, Vijay Kumar, Josyula R. Rao, and Pankaj Rohatgi. Efficient RJINDAELencryption implementation with composite field arithmetic. In *CHES2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 171–184. Springer, 2001.

[15] A. Satoh, S. Morioka, K. Takano, and Seiji Munetoh. A compact RJINDAELhardware architecture with S-box optimization. In *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 239–254. Springer, 2001.

[16] I.J. Cox, M.L. Miller and A.L. McKellips, *Watermarking as Communication with Side Information*, in Proc. IEEE, vol. 87, No. 7, pp. 1127–1141, 1999.

[17] M.L. Miller, G.J. Do¨err and I.J. Cox, *Applying Informed Coding and Embedding to Design a Robust High-Capacity Watermark*, IEEE Trans. Image Processing, vol. 13, No. 6, pp. 792–807, 2004.

[18] A. Abrardo and M. Barni, *Informed Watermarking by Means of Orthogonal and Quasi-Orthogonal Dirty Paper Coding*, IEEE Trans. Signal Processing, vol. 53, No. 2, pp. 824–833, 2005.

[19] I.J. Cox, J. Kilian, F.T. Leighton and T. Shamoon, *Secure Spread Spectrum Watermarking for Multimedia*, IEEE Trans. Image Processing, vol. 6, Issue 12, pp. 1673–1687, dec. 1997.

[20] M.L. Miller, I.J. Cox and J. Bloom, *Informed Embedding Exploiting Image and Detector Information during Watermark Insertion*, in Proc. IEEE Intl. Conference on Image Processing, ICIP'00, vol. III, pp. 1–4, 2000.

[21] S. Arora, L. Babai, J. Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. In Proceedings of the 1993 IEEE 34th Annual Foundations of Computer Science, pages 724{733. IEEE Computer Society, 1993.

[22] E Biham, A Biryukov, \An Improvement of Davies' Attack on DES", in *Journal of Cryptology* v 10 no 3 (Summer 97) pp 195{205

[23] E Biham, *How to Forge DES-Encrypted Messages in* 228 *Steps*, Technical Report CS884, Technion, August 1996