International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-1, Issue-6, January 2013

Investigation on Advanced Encryption Standard Techniques Using SMART Copyback for Data

Rajashekarappa, K M Sunjiv Soyjaudah

Abstract*: Advanced Encryption Standard (AES) is a symmetric block cipher that is intended to replace DES as the approved standard for a wide range of applications. Advanced Encryption Standard (AES) is the current standard for secret key encryption. NIST selected Rijndael as the proposed AES algorithm. The two researchers who developed and submitted Rijndael for the AES are both cryptographers from Belgium: Dr. Joan Daemen and Dr. Vincent Rijmen. Ultimately the Federal Information Processing Standard (FIPS PUB 197) used a standardized version of the algorithm called Rijndael for the Advanced Encryption Standard. The algorithm uses a combination of Exclusive-OR operations (XOR), octet substitution with an S-box, row and column rotations, and a MixColumn. It was successful because it was easy to implement and could run in a reasonable amount of time on a regular computer. In this paper we Investigate on Advanced Encryption Standard Techniques with some applications like Military, finance sector and maintain the research data for long time without fail. Finally these data could be store for long time with help of Self Monitoring Analysis and Reporting Technology (SMART) Copyback technique.

Keywords: Advanced Encryption Standard, Key search space.

I. INTRODUCTION

The most widely used encryption scheme is based on the Data Encryption Standard(DES) adopted in 1977 by the National Bureau of Standards, now the National Institute of Standards and Technology (NIST), as Federal Information Processing Standard46 (FIPS PUB 46) [1]. On January 2, 1997 the National Institute of Standards and Technology (NIST) held a contest for a new encryption standard. The previous standard, DES, was no longer adequate for security. It had been the standard since November 23, 1976. Computing power had increased a lot since then and the algorithm was no longer considered safe. In 1998 DES was cracked in less than three days by a specially made computer called the DES cracker[1].

Current alternatives to a new encryption standard were Triple DES (3DES) and International Data Encryption Algorithm (IDEA). The problem was IDEA and 3DES were too slow and IDEA was not free to implement due to patents. NIST wanted a free and easy to implement algorithm that would provide good security. Additionally they wanted the algorithm to be efficient and flexible[2][6].

After holding the contest for three years, NIST chose an algorithm created by two Belgian computer scientists, Vincent Rijmen and Joan Daemen. They named their algorithm Rijndael after themselves [2].

The rest of the paper is organized as follows: Section 2 presents the Advanced Encryption Standard (AES). Section 3 gives the Application of AES, Section 4 gives the Self Monitoring Analysis and Reporting Technology (SMART) Copyback and Section 5 gives the Experimental results are discussed. Section 6 concludes the paper.

II. ADVANCED ENCRYPTION STANDARD

Dr. Joan Daemen and Dr. Vincent Rijmen urbanized a block cipher called Rijndael. In AES the span of each block and the key can be autonomously specified to be 128, 192, or 256 bits. Before applying the algorithm to the data, the block and key sizes must be determined. AES allows for block sizes of 128, 168, 192, 224, and 256 bits. AES allows key sizes of 128, 192, and 256 bits [2]. The standard encryption uses AES-128 where both the block and key size are 128 bits. The block size is commonly denoted as N_b and the key size is commonly denoted as N_k. N_b refers to the number of columns in the block where each row in the column consists of four cells of 8 bytes each for AES-128 [3].

The following example will show how data is broken up into blocks[4]. Using AES 128 means that each block will consist of 128 bits. N_b can be calculated by dividing 128 by 32. The 32 comes from number of bytes in each column. In this case, N_b is 4. The original plain text is stored in bytes in a block. For example, the text “The true crypt” will be stored in a block as shown below in Figure 1**.**

Block

	0	1	2	3
0	T	t		p
1	h	r	c	t
2	e	u	r	
3		e	y	

Each character is stored in a cell of the block. The blank cells shown in the diagram are not really blank as they represent the spaces in the text. Depending on how the algorithm is implemented the characters may be stored as integer values, hexadecimal values, or even binary strings. All three ways represent the same data. Most diagrams show the hexadecimal values, however integer and string manipulation is much easier to do when actually programming AES. Figure 1 shows the values as characters for demonstration purposes to show how the text is stored into the block.

Revised Manuscript Received on 30 January 2013.
* Correspondence Author
Rajashekarappa*, Dept. of Computer Science and Engineering, JSSATE,Mauritius. (Research Scholar, Jain University, Bangalore), India.
Dr. K M Sunjiv Soyjaudah, Dept. of Electrical and Electronic Engineering,, University of Mauritius, Reduit, Mauritius, India.

The plain text is stored into blocks column by column and block by block until all the data is stored. In the example used above there were exactly 16 characters used for simplicity. In order to use the Rijndael algorithm the data must be a multiple of the block size, since all blocks need to be complete. When the data is not a multiple of the block size some form of padding must be used. Padding is when extra bits are added to the original data. One forms of padding includes adding the same bytes until the desired size is reached. Another option is padding with all zeros and having the last byte represent the number of zeros. Padding with null characters or random characters are also forms of padding that can be used [5][8].

Once a form of padding is chosen the data is represented as some number of complete blocks. The last thing needed before using the algorithm is the key. The key also known as the cipher key is also the same size as the block in this example. Unlike most data and transformations the cipher key can have any values chosen by the designer with no restrictions as long as the key is the correct length. The key is also stored as a block similar to the plain text.

When the plain text data is stored into blocks and the key is chosen the Rijndael encryption algorithm can be applied. Some of the following steps could be done before the encryption process starts, but for simplicity they will be discussed when they are needed.

- AES Rounds

This section briefly gives the overview of a basic level the Rijndael algorithm uses a number of rounds to transform the data for each block. The number of rounds used is 6 + the maximum of N_b and N_k. Following from the previous example of AES-128, the number of rounds is 10. This is calculated from 6 plus the maximum of (4,4). Since N_b and N_k are both 4, the number of rounds is 6 + 4 = 10 [2]. The initial block (also known as a state) is added to an expanded key derived from the initial cipher key. Then the round processing occurs consisting of operations of the S-box, shifts, and a MixColumn. The result state is then added to the next expanded key. This is done for all ten rounds, with the exception of the MixColumn operation of the final round. The final result is the encrypted cipher block[9].

A. Key Generation

The original cipher key needs to be expanded from 16 bytes to 16 (r + 1) bytes. In the example, there are ten rounds so r = 10. A round key is needed after each round and before the first round. Each round key needs to be 16 bytes because the block size is 16 bytes. Therefore, the cipher key needs to be expanded from 16 bytes to 16 (r + 1) bytes or 176 bytes. The expanded key is then broken up into round keys. Round keys are added to the current state after each round and before the first round. The details on the key expansion algorithm are complex and will be skipped. [4]

S-box

The first step to a round is to do a byte by byte substitution with a lookup table called an S -box. An S-box is a one to one mapping for all byte values from 0 to 255. The S-box is used to change the original plain text in bytes to cipher text[9]. The S-box is shown below in Figure 2. All values are represented in hexadecimal notation. This is how the S-box is commonly viewed. [2].

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Figure 2. S-Box.

For example if we had the plain text of one character "D" it would translate to the hexadecimal value of 44 by using an ASCII lookup table. When using the S-box the first digit in hexadecimal represents the rows of the table or the values on the left side going down. The second digit represents the column of the S- box, which, in this example, is also 4. Using the S-box we find row number 4 and column number 4 and find that the hexadecimal value in that cell is "1b". This is how the S-box works. Using this method with all the plain text will generate the new hexadecimal values that are used later in the algorithm.

The obvious question is: where did the S-box come from? This goes into modular arithmetic and a Galois field. A Galois field is a field with a finite number of elements. The Galois field is always a field that is a power of a prime. For each prime number there exists exactly one Galois field. The notation to represent a Galois field is GF(p), where p is the prime number[10].

For the S-box, the field GF(2^8) was chosen. There are several reasons to why this field was chosen. One obvious reason is that the power of 8 was chosen because there are 8 bits in a byte. The prime 2 was chosen because binary is represented as two possible digit's a 1 or a 0. In addition arithmetic is simple to do in this field because addition and subtraction are redefined as the exclusion or (XOR) operation. Invertability and resistance to algebraic attacks were also considered when forming the S-box.

To actually generate the S-box from the chosen field requires much more work. Careful thought was put into the transformation for security purposes. Using plain text for the next steps of the algorithm would make it more vulnerable so a byte substitution in the form of the S-box was used. The original bytes are transformed by using the multiplicative inverse and an affinity matrix shown in Figure 3 to make the cipher text resistant to algebraic attacks[2].

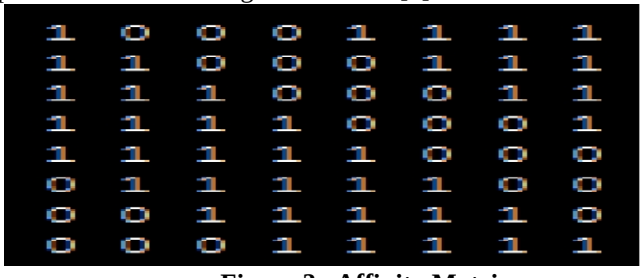

Figure 3. Affinity Matrix.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00	00	01	8d	f6	cb	52	7b	d1	e8	4f	29	c0	b0	e1	e5	c7
10	74	b4	aa	4b	99	2b	60	5f	58	3f	fd	cc	ff	40	ee	b2
20	3a	6e	5a	f1	55	4d	a8	c9	c1	0a	98	15	30	44	a2	c2
30	2c	45	92	6c	f3	39	66	42	f2	35	20	6f	77	bb	59	19
40	1d	fe	37	67	2d	31	f5	69	a7	64	ab	13	54	25	e9	09
50	ed	5c	05	ca	4c	24	87	bf	18	3e	22	f0	51	ec	61	17
60	16	5e	af	d3	49	a6	36	43	f4	47	91	df	33	93	21	3b
70	79	b7	97	85	10	b5	ba	3c	b6	70	d0	06	a1	fa	81	82
80	83	7e	7f	80	96	73	be	56	9b	9e	95	d9	f7	02	b9	a4
90	de	6a	32	6d	d8	8a	84	72	2a	14	9f	88	f9	dc	89	9a
a0	fb	7c	2e	c3	8f	b8	65	48	26	c8	12	4a	ce	e7	d2	62
b0	0c	e0	1f	ef	11	75	78	71	a5	8e	76	3d	bd	bc	86	57
c0	0b	28	2f	a3	da	d4	e4	0f	a9	27	53	04	1b	fc	ac	e6
d0	7a	07	ae	63	c5	db	e2	ea	94	8b	c4	d5	9d	f8	90	6b
e0	b1	0d	d6	eb	c6	0e	cf	ad	08	4e	d7	e3	5d	50	1e	b3
f0	5b	23	38	34	68	46	03	8c	dd	9c	7d	a0	cd	1a	41	1c

Figure 4. Multiplicative Inverse Table GF(2^8).

To calculate the multiplicative inverse is complicated and too much of a digression; therefore, a given table will be used instead. The field of all the multiplicative inverses is shown in Figure 4.

Using an example of the hexadecimal value "31" from the multiplicative inverse table we get the value "45" from the table. This is the value we will use. Our affinity matrix is in binary therefore the number must be in binary as well. 45h is 0100 0101 in binary. An easier way to view the matrix multiplication is as polynomial multiplication. The value 0100 0101 can be represented as the polynomial $0x^7 + 1x^6 + 0x^5 + 0x^4 + 0x^3 + 1x^2 + 0x^1 + 1$. This can be rewritten and simplified as $x^6 + x^2 + 1$. Taking the first row of the affinity matrix we can assign each bit a variable. Therefore, 1 0 0 0 1 1 1 1 can be represented as $N_0\ N_1\ N_2\ N_3\ N_4\ N_5\ N_6\ N_7$. This is generalized to all the rows, so all rows can be represented with the same variables in the same order. To do the multiplication start with the first row and see which variables are a 1 in the affinity matrix. Since the first row is 1 0 0 0 1 1 1 1, we only have to worry about the values $N_0\ N_4\ N_5\ N_6\ N_7$. Next, we look at the polynomial $x^6 + x^2 + 1$. The powers of this polynomial are what correspond to the subscripts of N[5]. Wherever the power in the polynomial is also in the subscript of N we can assign a value of 1. Otherwise we assign the value of 0. This makes $N_0\ N_4\ N_5\ N_6\ N_7$ change to 1 0 0 1 0. A further shortcut can be taken to ignore all 0's completely. This can be done because addition in the field GF(2^8) is just the XOR operation. Therefore, adding an even number of 1's will result in a value of 0 and adding an odd number of 1's will result in a value of 1. Shown below in Figure 5 is the simplified mathematics for the entire affine transformation.

$$\text{Row } 1 = 1\,0\,0\,0\,1\,1\,1\,1 = N_0 + N_4 + N_5 + N_6 + N_7 \quad 1+0+0+1+0=0$$

$$\text{Row } 2 = 1\,1\,0\,0\,0\,1\,1\,1 = N_0 + N_1 + N_5 + N_6 + N_7 \quad 1+0+0+1+0=0$$

$$\text{Row } 3 = 1\,1\,1\,0\,0\,0\,1\,1 = N_0 + N_1 + N_2 + N_6 + N_7 \quad 1+0+1+1+0=1$$

$$\text{Row } 4 = 1\,1\,1\,1\,0\,0\,0\,1 = N_0 + N_1 + N_2 + N_3 + N_7 \quad 1+0+1+0+0=0$$

$$\text{Row } 5 = 1\,1\,1\,1\,1\,0\,0\,0 = N_0 + N_1 + N_2 + N_3 + N_4 \quad 1+0+1+0+0=0$$

$$\text{Row } 6 = 0\,1\,1\,1\,1\,1\,0\,0 = N_1 + N_2 + N_3 + N_4 + N_5 \quad 0+1+0+0+0=1$$

$$\text{Row } 7 = 0\,0\,1\,1\,1\,1\,1\,0 = N_2 + N_3 + N_4 + N_5 + N_6 \quad 1+0+0+0+1=0$$

$$\text{Row } 8 = 0\,0\,0\,1\,1\,1\,1\,1 = N_3 + N_4 + N_5 + N_6 + N_7 \quad 0+0+0+1+0=1$$

Figure 5 Simplified Multiplication example. Rows are multiplied by the polynomial x^6+x^2+1.

The result of the multiplication is the vector 0 0 1 0 0 1 0 1. The final step to the transformation is to add the vector 1 1 0 0 0 1 1 0 to our vector. This gives us the final binary string of 1 1 1 1 0 0 0 1 1. Since we want the last row represented as the first bit we need to reverse this string giving us the new binary string of 1 1 0 0 0 1 1 1. In hexadecimal this value is C7.

Rijndael Decryption

Decryption occurs through the function AddRoundKey(), plus the inverse AES functions InvShiftRows(), InvSubBytes(), and InvMixColumns(). AddRoundKey() does not require an inverse function, as it simply XORs the state with the subkey (XOR encrypts when applied once, and decrypts when applied again).

Decryption is simple after understanding the encryption process. It is basically just the inverse. The algorithm was designed for all the steps to be invertible so decryption is basically like doing everything backwards. Therefore, for decryption starts at the last round and the last round key. When processing each round do the process backwards. So, the round key is added first to the last round. Addition is its own inverse, which is nice. Then the MixColumn step is applied. The MixColumn step is applied to all rounds except the last one. Also the inverse MixColumn table is used[2]. This table is generated with another matrix similar to the way the MixColumn table was generated. The difference is that there are no short cuts to generate the table. Therefore, the matrix multiplication needs to be performed in the field GF(2^8)[5].

III. APPLICATION OF AES

Since the minimum key size of AES is 128-bits, it is considered to be immune to brute force attacks for the near future. Given the strength of the cipher, implementing AES requires relatively few resources in terms of memory and system cycles, which makes it a good choice for an encryption algorithm. Some sample applications where AES is useful are:

- Wireless communication, such as wireless keyboards.
- Point-of-sale terminals,
- Surveillance applications

The Advanced Encryption Standard (AES) is an algorithm used to encrypt and decrypt data for the purposes of protecting the data when it is transmitted electronically. This kind of secretes information transforming between From A to B, for example in Military Application. Similarly in financial sector and research field.

The AES algorithm allows for the use of cipher keys that are 128-, 192-, or 256bits long to protect data in 16byte blocks.

AES is a U.S. Federal Information Processing Standards approved algorithm that is also approved for commercial and private applications.

Since its acceptance in 2001, AES has become widely used in a variety of applications. The AES algorithm is a reduced version of the Rijndael algorithm, though the names are sometimes used interchangeably. The Rijndael algorithm allows for additional key sizes and data sizes that are not supported by AES.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

IV. SELF MONITORING ANALYSIS AND REPORTING TECHNOLOGY COPYBACK

In this paper users have variety of requirements of data storage that can be addressed using Self Monitoring Analysis and Reporting Technology (SMART) Copyback [7]. It is estimated that over 94% of all new information produced in the world is being stored on magnetic media, most of it on Physical Disks (PD). Despite their importance, there is relatively little published work on the failure patterns of Physical Disks (PD), and the key factors that affect their lifetime. Most available data are either based on extrapolation from accelerated aging experiments or from relatively modest sized field studies. Moreover, larger population studies rarely have the infrastructure in place to collect health signals from components in operation, which is critical information for detailed failure analysis. It presents the data collected from detailed observations of a large disk drive population in a production Internet services deployment. The population observed is many times larger than that of previous studies. In addition to presenting failure statistics and analyze the correlation between failures and several parameters generally believed to impact longevity. Analysis identifies several parameters from the Physical Disks (PD), self monitoring facility (SMART) that correlate highly with failures. Despite this high correlation conclude that models based on SMART parameters alone are unlikely to be useful for predicting individual drive failures. Surprisingly, it found that temperature and activity levels were much less correlated with Physical Disk (PD) failures

In this study we report on the failure characteristics of consumer-grade Physical Disks (PD). Analysis is made possible by a new highly parallel health data collection and analysis infrastructure, and by the sheer size of our computing deployment.

Our results confirm the findings of some of the SMART Copyback parameters are well-correlated with higher failure probabilities. First errors in reallocation, offline reallocation, and probational counts are also strongly correlated to higher failure probabilities. Despite those strong correlations, we find that failure prediction models based on SMART Copyback parameters alone are a likely to be severely limited in their prediction accuracy, given that a large fraction of our failed Physical Disks have shown no SMART error signals whatsoever. This results suggests that SMART Copyback models are more useful in predicting trends for large aggregate populations than for individual components[7]. It also suggests that powerful predictive models need to make use of signals beyond those provided by SMART Copyback. In this thesis we report on the failure characteristics of consumer-grade Physical Disks (PD). The drive vendors builds a logic in to the drives to make drives smart so that the user gets warning signal as a “predictive failure” whenever the drive is about to go bad for some reason. The drives built with this kind of logic are called a “SMART” drive which is an acronym for “Self-Monitoring Analysis and Reporting Technology”. Using this technique and make use of cryptanalysis data keeping for long time without fail.

V. EXPERIMENTAL RESULTS

This section includes the some example input vectors specification for the input data and cipher keys. All values are presented in hexadecimal format.

128-Bit Cipher Key:

Input Data: 0x00112233445566778899AABBCCDDEEFF
Cipher Key: 0x000102030405060708090A0B0C0D0E0F
EncryptedData:0x69C4E0D86A7B0430D8CDB78070B4C55A

192-Bit Cipher Key

Input Data: 0x00112233445566778899AABBCCDDEEFF
CipherKey:0x000102030405060708090A0B0C0D0E0F1011121314151617
EncryptedData:0xDDA97CA4864CDFE06EAF70A0EC0D7191

256-Bit Cipher Key

Input Data:0x00112233445566778899AABBCCDDEEFF
CipherKey:
0x000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
Encrypted Data:
0x8EA2B7CA516745BFEAFC49904B496089

The proper execution of the encrypt and decrypt routines can be confirmed in the firmware by setting a break point after the InvCipher() function, which is called in AES_Main.c. Add the variables, EncryptedData and PlaintextData, to the IDE watch window and confirm that their values are the same as the ones listed above. Plaintext Data should be equivalent to Input Data.

SMART Copyback Experimental Results

The experiments were implemented in C language on a Pentium IV(1.83 Ghtz). Many more experiments are conducted using the following Hardware and Softwares:

- Pentium Processors
- 1 GB RAM
- 1068 PCI Controller Card
- PCI-E Slots on Motherboard
- SATA/SAS Hard Disks

Software requirements

- Operating System: Windows 2003(32 bit/64 bit)
- Tera Term Pro Web Version 3.1.3
- Visual C 6.0
- Mega Storage Management (MSM) Version 2.9

Other Software Requirements

- MegaCli

Unit Test Cases:

The following table shows the details of the every test case with description, input, expected output and actual result appeared.

1. Test Case: UTC-01

The test case UTC-01 given below is the Creation of RAID1 using Two Physical Disks (PDs). It will take the input from the user and creates the RAID1 and generates a log file. As the below Table 5.1 illustrate the Test Case for virtual disk creation success test.

Table 5.1 Test Case for Virtual Disk Creation Success Test

SI No. of Test Case	**UTC -01**
Name of test	***Virtual Disk Creation Success test***
Feature being tested	*Creating a RAID1 using Two* Physical Disks (PD)

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

Sample Input	*Select Physical Disks (PDs)*
Expected Output	*Action Performed and its event log filed*
Actual output	*As Expected*
Remarks	*Test Successful*

2. Test Case: UTC-02

The test case UTC-02 given below is the Creation of RAID 5 using Three Physical Disks (PD). It will take the input from the user and creates the RAID5 and generates a log file. As the below Table 5.2 illustrate the Test Case for virtual disk creation success test.

Table 5.2 Test Case for Virtual Disk Creation Success Test

SL.No. of Test Case	**UTC -02**
Name of test	Virtual Disk Creation Success test
Feature being tested	Creating a RAID5 using Three Physical Disks (PD)
Sample Input	Select Physical Disks (PDs)
Expected Output	Action Performed and its log file
Actual output	As Expected
Remarks	Test Successful

3. Test Case: UTC-03
Test case Manual Copyback

This test case is illustrated in below Table 5.3. This test case is used for Manual Copyback module to test the functionality of the function which writes data from Source Physical Disk (PD) to Destination Physical Disk (PD). Check Physical Disk (PD) Allowed Operations.
Expected Result :
startCopyback should be TRUE for all ONLINE PDs
stopCopyback should be TRUE for all PDS_COPYBACK PDs.
Start copyback manually from ONLINE to Unconf. Good/HSP.
Expected Result :After copyback completes, the source would become Unconf. Good or Unconf. Bad(if it has predictive failure). The destination would become ONLINE.
Note that copyback can never be started to a destination HSP which already has a predictive failure.
Stop copyback on PDS_COPYBACK PDs.

Table 5.3 Test Case for Manual Copyback – failure test

Serial No. of test case	**UTC-03**
Name of the Test	Manual Copyback – failure test
Item/Feature being Tested	Check Physical Disk (PD) Allowed Operations.
Sample Input	Manual Copyback module to test the functionality of the function which writes data from Source Physical Disk (PD) to Destination Physical Disk (PD). Check Physical Disk (PD) Allowed Operations.
Expected output	startCopyback should be TRUE for all ONLINE PDs stopCopyback should be TRUE for all PDS_COPYBACK PDs Start copyback manually from ONLINE to Unconf. Good/HSP
Actual output	After copyback completes, the source would become Unconf. Good or Unconf. Bad(if it has predictive failure). The destination would become ONLINE. Note that copyback can never be started to a destination HSP which already has a predictive failure. Stop copyback on PDS_CO PYBACK PDs.
Remarks	The Hotspare(HSP) size should be more than an array size or equal to an Array size. As this condition is failed Because destination HSP size is less than an Array size

VI. CONCLUSION

Usually lightweight encryption algorithms are very attractive for many applications. For the security of block data, we found an encryption algorithm that is based on AES using symmetric key encryption algorithm. The Rijndael algorithm was chosen as the new Advanced Encryption Standard (AES) for several reasons. The purpose was to create an algorithm that was resistant against known attacks, simple, and quick to code. Choosing to use field $GF(2)^8$ was a very good decision. The inverse of the addition operation was itself, making much of the algorithm easy to do. In fact, every operation is invertible by design. In addition, the block size and key size can vary making the algorithm versatile. As of now, no practical attacks have been successful on AES. The Advanced Encryption Standard data could be store for long time with help of Self Monitoring Analysis and Reporting Technology (SMART) Copyback technique.

REFERENCES

1. Rajashekarappa and Dr. K M S Soyjaudah "Heuristic Search Procedures for Cryptanalysis and Development of Enhanced Cryptographic Techniques" Published at International Journal of Modern Engineering Research (IJMER), May 2012, Vol.2, Issue.3, pp-949-954.
2. William Stallings, "Cryptography and Network Security Principles and Practices", Third edition, McGraw- Hill, 2006.M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.
3. Announcing the ADVANCED ENCRYPTION STANDARD (AES), Federal Information Processing Standards Publication 197 November 26, 2000.
4. Advanced Encryption Standard (AES). FIPS. November 23, 2001. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
5. Behrouz A. Forouzan, "Cryptography and Network Security", Firstedition, McGraw- Hill, 2006.
6. Kaufman, C., Perlman, R., and Speciner, M. Network Security: Private Communication in a Public World. 2nd Ed. Upper Saddle River, N. J.:Prentice Hall PTR, 2002.
7. Rajashekarappa and Dr. K M S Soyjaudah "Self Monitoring Analysis and Reporting Technology (SMART) Copyback", proceedings of International Conference on Information Processing 2011 (ICIP 2011), 8th-9th Aug, Bangalore, pp 463 - 469, 2011. © Springer-Verlag Berlin Heidelberg, 2011.
8. Data Encryption Standard, Federal Information Processing Standards Publication No. 46, National Bureau of Standards, January 15, 1977.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

9. Krishnamurthy G N, V Ramaswamy, "Making AES Stronger: AES with Key Dependent S-Box", IJCSNS International Journal of Computer Science and Network Security, Vol.8, No.9, pp 388-398, September 2008.
10. P. Noo-intara, S. Chantarawong, and S. Choomchuay "Architectures for MixColumn Transform for the AES" Department of Electronics, Faculty of Engineering, and Research Center for communications and Information Technolohy (ReCCIT) King Nongkut's Institute of Technology Ladkrabang (KMITL), Bangkok 10520, Thailand.

AUTHOR PROFILE

Mr. Rajashekarappa is a Lecturer since July 2010 in the Department of Computer Science and Engineering, JSS Academy of Technical Education, Avenue Droopnath Ramphul, Bonne Terre, Vacoas, Mauritius. He has one and half years of experienced as a Project Assistant at Indian Institute of Science (IISc), Bangalore, India. He worked as Project Internee in Indian Space Research Organization (ISRO), Bangalore, India. He has one and half years of experienced as a Project Trainee at LSI Technologies Pvt. Ltd, Bangalore, India.

Mr. Rajashekarappa obtained his Bachelor of Engineering in Computer Science and Engineering from Anjuman Engineering College, Bhatkal, India. He has qualified in Graduate Aptitude Test in Engineering (GATE), Computer Science and Engineering, 2006. He received his Master Degree in Computer Science and Engineering from R. V. College of Engineering, Bangalore, India. He is pursuing his Ph. D in Computer Science and Engineering at Jain University, Bangalore, India, under the guidance of Dr. K M Sunjiv Soyjaudah, Professor ,University of Mauritius, Reduit, Mauritius. Mr. Rajashekarappa area of interest and research include Cryptography, Data mining, Mobile Communication, Computer Networks and Cloud Computing. He has published several Research papers in international journal/conferences. He has guided many students of Bachelor degree in Computer Science and Engineering in their major projects. Mr. Rajashekarappa is a member of ISTE, IETE, IACSIT, IAEST, IAENG and AIRCC.

Professor K M Sunjiv Soyjaudah received his B. Sc (Hons) degree in Physics from Queen Mary College, University of London in 1982, his M.Sc. Degree in Digital Electronics from King's College, University of London in 1991, and his Ph. D. degree in Digital Communications from University of Mauritius in 1998. He is presently Professor of Communications Engineering in the Department of Electrical and Electronic Engineering of the University of Mauritius. His current interest includes source and channel coding modulation, cryptography, voice and video through IP, as well as mobile communication. Professor K M Sunjiv Soyjaudah is a member of the IEEE, Director in the Multicarrier (Mauritius), Technical Expert in the Energy Efficiency Management Office, Mauritius.