Communications on Applied Electronics (CAE) – ISSN : 2394-4714
Foundation of Computer Science FCS, New York, USA
Volume 7 – No. 40, January 2025 – www.caeaccess.org

Creating a Versatile S-Box for an N-bit AES Algorithm using Reversible Logic

Rohini S.H.
Electronics and Communication
KLE Technological University, Hubli, India

Rajashekar B. Shettar
Electronics and Communication
KLE Technological University, Hubli, India

Supriya K.
Electronics and Communication
KLE Technological University, Hubli, India

Rajeshwari M.
Electronics and Communication
KLE Technological University, Hubli, India

ABSTRACT
In the state of art technology, substantial amount of research on quantum computers is explored. Mathematical problems unsolved by present day computers are well addressed by quantum device, such future computers are capable of breaking existing public key crypto-systems very well. Any cryptographic technique's strength depends on the key size and secure transmission. To overcome the security issues and attacks by future computers, we propose quantum resistant and generalized approach to AES(Advanced Encryption Standard). AES is the most widely used symmetric key and block cipher cryptographic algorithm, known for its high security, speed, and strength. According to Shannon's theorem, achieving optimal security requires a one-to-one correspondence between each bit of the message and each bit of the key, meaning that the lengths of both the key and the message should be the same. Hence, we proposed an approach , where message size is increased from 128 bits to 192, 256, 512 and 1024 bits and key size is also made same as that of message size. The security of AES algorithm is enhanced by creating dynamic S-Box, as S-Box used in standard AES is static throughout AES encryption. The confusion capability for AES is provided by this S-Box. The S-Box used in standard AES is static throughout the encryption process. In order to make AES more secure, dynamic S-Box is created. This dynamic S-Box is made dependent on message and key. Each time S-Box varies according to message and key. The implementations of AES with message sizes of 192, 256, 512, 1024 and key sizes as that of message size are discussed. Dynamic S-Box is proposed and implemented using reversible logic to mitigate the power dissipation. The proposed scheme is simulated and the analysis of power is carried out using the Genus tool, proving it to be efficient in terms of power, gate usage, garbage, and quantum cost. Testing of dynamic S-Box is done using hamming distance, strict avalanche effect, correlation factor.

Keywords
AES, S-Box, Key, Dynamic S-Box.

1. INTRODUCTION
With recent advances in VLSI Technology, it is possible to integrate billions of devices on the chip. The rapidly growing technology and the rise in the number of circuits on the chip and off-chip demands for reduction of power dissipation. The reduction of power happens to be one of the essential goals in the VLSI design for many years. The reversible logic circuits are the ones that play an essential role in the design of low power circuits for future computers. Research interest in the existing area evolved, when Landauer emphasized on power dissipation, which was observed after implementation of conventional binary irreversible gate [1]. This implies that integration of computation in irreversible order yields KTln2 joules heat for loss of information per bit. It is further evidenced by the approach of thermodynamics that assignment of reversible computation nullifies the heat generation [2]. K. S. Nag et.el [3] realized AES encryption and decryption on FPGA. The design is synthesized and verified using Xilinx. The implementation is done for AES with message and key size of 128 bits.

The paper [4] proposes the Advanced Encryption Standard (AES-192) with multiple keys, implemented using a Field Programmable Gate Array (FPGA).

E.-N. Mui [5] explored a combinational logic-based S-Box for the SubByte transformation, detailing its internal operations. This implementation offers higher speeds compared to traditional ROM-based lookup tables, as it can be pipelined and occupies less area..

E. M. Mahmoud et.el [6] designed dynamic S-box for AES. The S-box designed is dependent on key. The dynamic S-box designed has same characteristics as that of original S-box of AES. The dynamic S-Box is designed to provide more confusion for original AES algorithm.

H. Prasad et.el [7], explored substitution table, multiplicative inverse and affine transform mathematics in Galois field to do Sub-byte transformation. An encryption table is used along with S-Box to encrypt the message.

G. Jacob et al. [8] propose a scheme for creating a dynamic key-dependent S-Box for the SubBytes transformation used in cryptographic techniques.

Pradeep L.N. et al. [9] proposed a method for generating random session keys, which are then used to create S-boxes. This random key generation helps mitigate brute force attacks, while the key-dependent S-box enhances the cipher's resistance to linear and differential cryptanalysis.

Z.-q. Du et.el [10] described the generation of dynamic S-Box. In which 8-bits key and 8-bits plaintext is XORed to generate the middle cipher text. The cipher text is divided into two parts that one is used for determining S-Box and another for substitution. S-Box is formed from combining 16 4x4 S-Boxes.

2. AES WITH EQUAL KEY AND MESSAGE SIZE ENCRYPTION

Shannon's theorem states that an individual correspondence between each bit of the message and each bit of the key would give the best security i.e, the length of the key must equal the length of the plaintext message and the key can be used only once [1]. According to Shannon's, for perfect secrecy, we must have ($K \geq P$), where K is key and P is plaintext. Hence, proposed work presents variation of the standard AES algorithm by increasing message, key size and also making both key, message sizes equal. For applications that demand enhanced security, regardless of area and computational time, the message size and key size of standard AES are increased, resulting in a more complex algorithm. Using equal key size as that of message size results in more security [4]. Our proposed work uses 192, 256, 512, 1024 bits block size and same size key and implemented using reversible logic.

AES encryption involves four transformations: Byte Substitution, Shift Rows, Mix Columns, and Round Key Addition. The block diagram of AES Encryption is shown in Figure 1. AES involves four steps which includes:

- S-Box Substitution Bytes
- Shift Row Transformation
- MixColumn Transformation
- AddRound Key

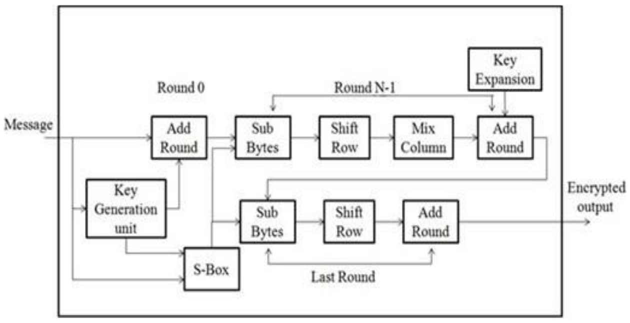

Fig. 1. Functional Diagram of AES Encryption.

3. PROPOSED DYNAMIC S-BOX

S-Box is used to introduce confusion while doing encryption , which is main core. In conventional AES, this S-Box is static throughout process. In order to make AES more secure, dynamic S-Box is created. This dynamic S-Box is made dependent on message and key. Each time S-Box varies according to message and key. The flow of the proposed work is shown in Figure 2. Procedure for implementing dynamic S-box is discussed in Algorithm 1. Algorithm is uniquely designed, to generate 256 S-box values dynamically every time, when incoming message changes.

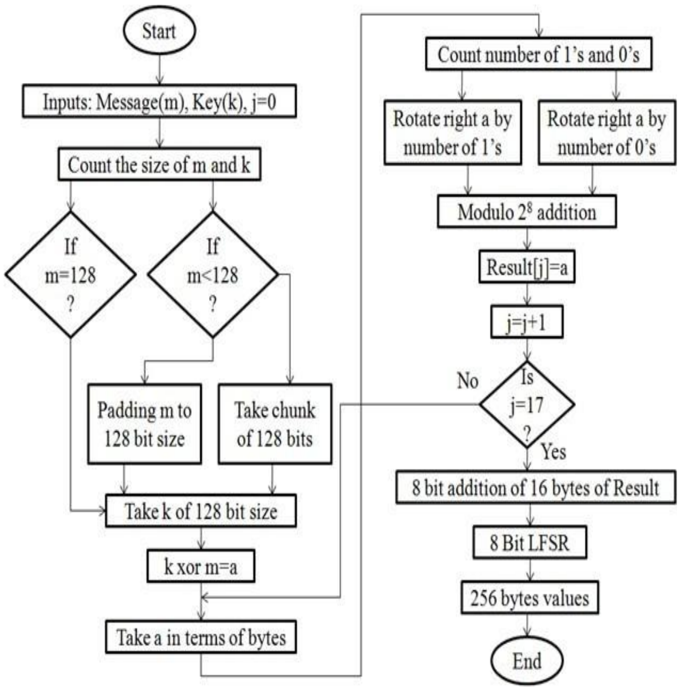
Start
Inputs: Message(m), Key(k), j=0
Count the size of m and k
If m=128 ?
If m<128 ?
Padding m to 128 bit size
Take chunk of 128 bits
Take k of 128 bit size
k xor m=a
Take a in terms of bytes
Count number of 1's and 0's
Rotate right a by number of 1's
Rotate right a by number of 0's
Modulo 2^8 addition
Result[j]=a
j=j+1
No
Is j=17 ?
Yes
8 bit addition of 16 bytes of Result
8 Bit LFSR
256 bytes values
End

Fig. 2. Flow chart of Proposed Dynamic S-Box.

The message and key can be of any size, but in our proposed algorithm the message and key of 128 bits are selected from the inputs. Modulo 2 addition operation is performed on 128 bit message and key using reversible gate i.e, Feynman gate, produces 128 bit output. It is divided into 16 bytes. From 16 bytes, the first byte is taken and the number of ones and zeros are calculated. The same byte is right rotated by the number of 1's as well as by the number of 0's. Thus two right rotated bytes are obtained. These two bytes are added using a modulo adder (2^8). The same procedure is repeated for all the remaining 15 bytes. Now, 16 bytes of results are obtained from the modulo adder. These 16 bytes are then added using a reversible adder to get one byte of data, neglecting the carry. The addition is performed using a reversible ripple carry adder. The reversible gate used in the ripple carry adder is the MTS gate. Thus, the obtained one byte of data from the ripple carry adder is given as an input to 8-bit LFSR. From the 8-bit LFSR, 255 non - repeated bytes are generated except 00. So, 256th byte is taken as 00. Thus, these 256 non-repeated values are generated and are used to substitute the values of the state matrix.

```
Algorithm 1: Dynamic S-Box
1  Input: Message m, Key value k
2  Output: S-Box of 256 values
3  Begin
4  Initialize Message m, Key k
5  if m=128 bits then
6      go to step 15
7  end
8  if m<128 bits then
9      Pad m with zeros to make its length 128 bits, go to step 15
10 end
11 else if m>128 then
12     go to step 14
13 end
14 Select chunk of 128 bits each
15 Calculate a= k?  m, ?here k=128 bits
16 Convert a in terms of 16 bytes ie. a[0] to a[15]
17   Count the number of 1's and 0's in a[n]
     Take 1's count in one[n] and 0's in zero[n]
     for i=0 to i=15 do
17     Calculate b[n]= ror(b[n],a[n])
18     Calculate c[n]= ror(c[n],a[n])
19     Calculate d[n]=b[n] ?  c[n] ie. modulo 2^8 addition
20   end
24 Calculate d'[n] ie d'=d[0]+d[1]+. ..........................+d[15]
25 Feed d' (a byte value) to 8 bit LFSR
26 Feed these 256 values(including zero) create dynamic S-Box
27 End
```

3.1 Implementation Details

3.1.1 LFSR (Linear Feedback Shift Register)

A LFSR produces an n-sequence i.e., it will cycle through all possible states in the shift register. In LFSR, the initial value is called as seed and because of deterministic operation of the register, the values produced by it is determined by its present or past state. After finite number of states, value will repeat in LFSR. The bits that connect to the next states are called the taps. The LFSR used has the taps [8,4,3,2,1] shown in Figure 3 and also the reversible implementation of 8 bit LFSR is shown in Figure 4. The taps are XOR'd and then given as an input to the rightmost shift register.

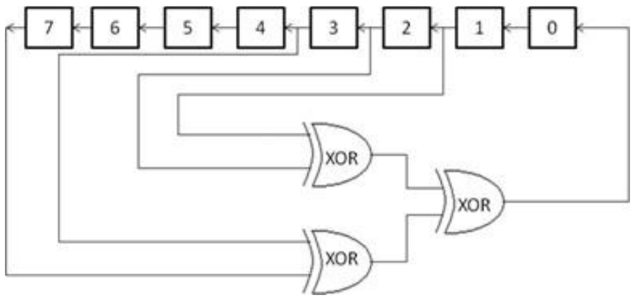

Fig. 3. 8-bit LFSR

Modulo addition

Figure 5 shows reversible logic based modulo Adder block. If the result of two 8 bit numbers is 9 bit , then the Modulo Adder is designed to make the result as 8 bit. If the carry is generated, then the result is subtracted from 256 (100000000)to make 9th bit of result as zero.

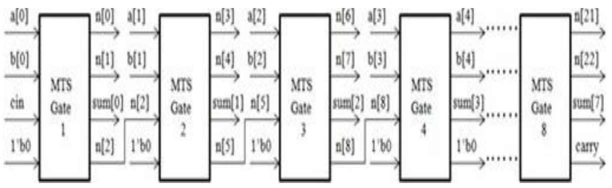

Fig. 4. Reversible diagram of 8 bit LFSR.

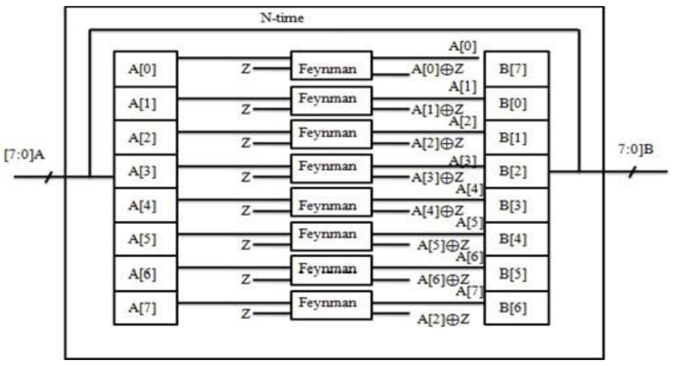

Fig. 5. Reversible design of Modulo Addition.

3.1.2 *Shift Row Transformation*

The shift row transformation of 8-bit data is shown in Figure 6. One bit is shifted cyclically to the left at a time. Row number is used to rotate each row in case of shift transformation. It is implemented using reversible logic gate Feynman. In general, 'N' is the amount to be shifted. The same procedure is followed for 32, 64 and 128, 256 bits data transformation in case of 192, 256, 512 respectively. Figure 7. shows reversible design of Mix-column Matrix of 2x2.

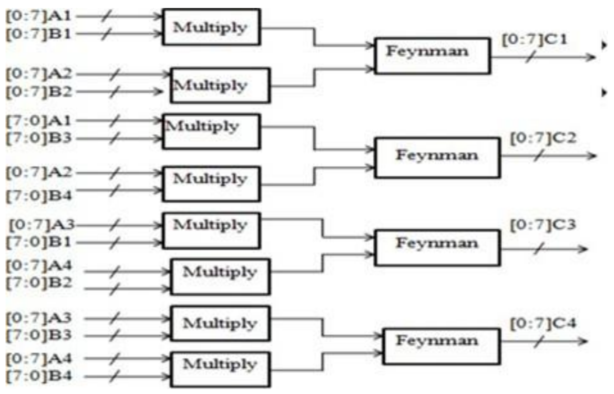

Fig. 6. Reversible design of Shift Row Transformation.

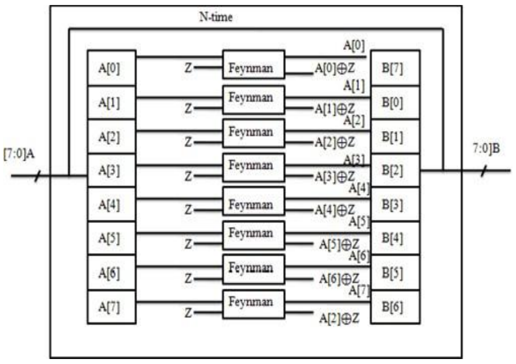

Fig. 7. Reversible design of Mix-column Matrix of 2x2

4. RESULTS AND DISCUSSIONS

Verilog coding was performed to verify the functionality of the proposed work, which was tested on a Virtex5 (ML505) FPGA using Chipscope. Since the S-Box used in AES is static, a dynamic S-Box is generated to enhance the strength of the AES algorithm. The new approach proposes generating the S-Box dynamically based on the message and key. The simulated result of the dynamic S-box is shown in Figure 8. All 256 bytes of S -Box are arranged in one dimensional array as follows: 67ce9c3973e7cf9e3c78f1e3c68d1b366cd8b061c28408112245 8b172e5dba75ebd
7af5ebd7bf6eddbb76fdfbe7cf8f0e1c3860d1a3469d3a64d9a35 6bd6ad5bb66dda

b56ad4a850a04182040913274e9d3b76ecd9b264c992244993 264c983060c081030
70e1d3a74e9d2a4489123478e1c3871e2c48810204080010205 0b162c58b163c78f

1e3d7af4e8d0a143870f1f3f7ffffefcf9f2e4c8902142850a1429 53a74f9f3e7dfaf5ea
d5aa55ab57ae5cb870e0c183060c183162c58a152b56ac59b36 6cc993265cb 972
f5fbf7efdfbf7efdebc79f3e6cd9b376eddbb77eedcb972e5ca952 a54a952a54a9428
51a2448912254b962d5ab468d1a3468c193300

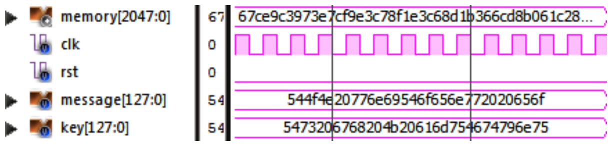
memory[2047:0]
67
67ce9c3973e7cf9e3c78f1e3c68d1b366cd8b061c28...
clk
0
rst
0
message[127:0]
54
544f4e20776e69546f656e772020656f
key[127:0]
54
5473206768204b20616d754674796e75

Fig 8. Simulation result of dynamic S-box

4.1 Test Results

Avalanche Effect due to 1- bit change in plaintext for AES 128 bit with proposed key unit and dynamic s-box is shown in Table 1, and it is on an average of 50% is achieved. Plot of same is shown in Figure 9, it indicates that proposed dynamic s-box is stable enough to maintain security level of AES.

Table 1. Avalanche effect due to 1 bit change in plain text

Bit variation index	Plain text	Cipher text	Avalanche effect (%)
127	544f4e20776e69546f656e772020656f	7b76d81919c48db922d570067c01c33a	50.7812
124	444f4e20776e69546f656e772020656f	46c0282f2055d8a5798a3ceffa599ad6	56.25
109	544f3e20776e69546f656e772020656f	d9142c80551793ca3a7180c3c6871232	53.125
75	544f4e20776e69547f656e772020656f	2dbaf27fc59d58d051abd356a35fa87d	55.4687
43	544f4e20776e69546f656f772020656f	d396dae0e5dda55a0e01a25aa514602d	42.1875
32	544f4e20776e69546f656e782020656f	60fb03313f8f422a74996e0cdbc298cd	50.0
12	544f4e20776e69546f656e772020756f	00650a54229637f9d2782f5c7779fe5a	50.0
2	544f4e20776e69546f656e772020656b	356f8ee7d99e3ba21ac50768c92dcdbf	46.875
1	544f4e20776e69546f656e772020656d	e4bbf9978af920f7f4b813830d92167d	58.5937
0	544f4e20776e69546f656e772020656e	3e7f64d0e4c6f642561d29cc1ca07914	46.875

Table 2. Comparison between proposed AES128 with dynamic S-box

Bit variation index	Plain text	Cipher text of Standard AES	Avalanche effect (%)
0	A07B00321C11759D0FDE340234384BC9	6EC1C564A5B556ED8DA6FF494B1942FE	49.2187
2	A07B00321C11759D0FDE340234384BCD	95B1FFBC83E62B7974079CC974906B1A	56.25
3	A07B00321C11759D0FDE340234384BC1	B0A172630ED34B8624EC94A5996FBCAE	50.0
13	A07B00321C11759D0FDE340234386BC9	F3E47315CB0CC4BA65437863A63EDC2A	53.125
40	A07B00321C11759D0FDE350234384BC9	1FDA4608CD964B68E369C2E65F63E647	60.1562
84	A07B00321C15759D0FDE340234384BC9	64D83BDF4BB4B8A013950F12B8454406	53.125
112	A07B00B21C11759D0FDE340234384BC9	144EA67A93D3DC5E46F049BFB1A3C05D	50.7812
121	A27B00321C11759D0FDE340234384BC9	351D0C6C16C89C55D4D36C6FE9150D99	49.2187
124	B07B00321C11759D0FDE340234384BC9	BEB94AF37F95ED11CFA3B1A3F60BB459	42.9687
125	807B00321C11759D0FDE340234384BC9	5EC53DF412699470424786D670F9F5AA	53.9062

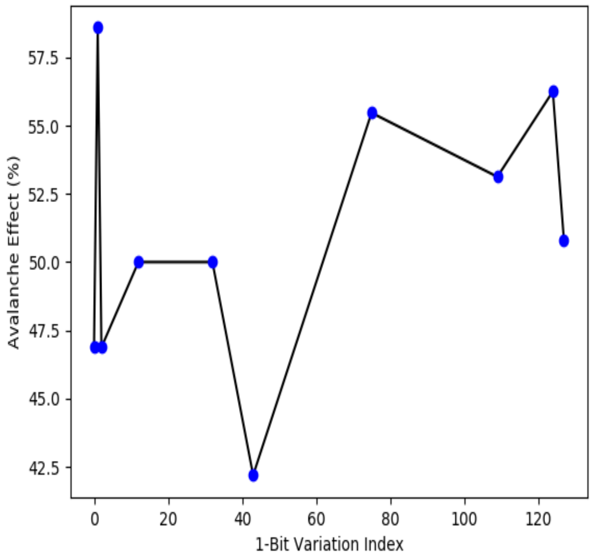

Fig 9. Avalanche Effect due to 1- bit change

Comparison between proposed AES128 with dynamic S-box and standard AES 128 with Key: B9B5ED7585C8B15D7454ED271AA3A3A3 is tabulated in Table 2. It shows that an average of 50% avalanche effect is observed and its well within the range set by NIST.

5. CONCLUSION

The security of AES algorithm is enhanced by creating dynamic S-Box, as S-Box used in standard AES is static throughout AES encryption. Dynamic S-Box is designed instead of using static S-Box. Proposed algorithm uses rotate operation and modulo addition, which helps to modify each and every bit of S-Box which creates more confusion as a part of SubByte. Work also includes 8 bit LFSR to ensure randomness in generated look table as a part of S-Box. This proposed algorithm leads to increase in the complexity and makes the differential and linear cryptanalysis more difficult. Avalanche test gives average of 50% , indicates that proposed algorithm is stable enough to maintain desired security level.

6. REFERENCES

[1] R. Landauer, "Irreversibility and Heat Generation in the Computational Process," IBM Journal of Research and Development", 5, pp. 183-191, 1961.

[2] C.H. Bennett, "Logical Reversibility of Computation,IBM J.Research and Development," pp. 525-532, November 1973.

[3] Y. Tsividis, "Operation and Modeling of The MOS Transistors"2nd McGraw Hill, 1999.

[4] S. U. Jonwal and P. P. Shingare, "Advanced encryption standard (AES) implementation on FPGA with hardware in loop," in Trends in Electronics and Informatics (ICEI), 2017 International Conference on, pp. 64–67, IEEE, 2017.

[5] K. S. Nag, H. Bhuvaneswari, and A. Nuthan, "Implementation of advanced encryption standard-192 bit using multiple keys," in IET Conference Proceedings, The Institution of Engineering & Technology, 2013.

[6] E.-N. Mui, "Practical Implementation of Rijndael S-Box Using Combinational Logic," [Online]. Available: http://www.xess.com/projects/ Rijndael_SBox. pdf, 2007.

[7] E. M. Mahmoud, A. Abd, E. Hafez, T. A. Elgarf, et al., "Dynamic AES-128 with key-dependent S-Box," 2013.

[8] H. Prasad, J. Kandpal, D. Sharma, and G. Verma, "Design of low power and secure implementation of S-Box for AES," in Computing for Sustainable Global Development (INDIACom), 2016 3rd International Conference on, pp. 2092–2097, IEEE, 2016.

[9] G. Jacob, A. Murugan, and I. Viola, "Towards the generation of a dynamic key- dependent S-Box to enhance security.," IACR Cryptology ePrint Archive, vol.2015, p. 92, 2015.

[10] Pradeep L.N., Bhattacharjya A. (2013) Random Key and Key Dependent S-box Gen- eration for AES Cipher to Overcome Known Attacks. In: Thampi S.M., Atrey P.K., Fan CI., Perez G.M. (eds) Security in Computing and Communications. SSCC 2013. Com- munications in Computer and Information Science, vol 377. Springer, Berlin, Heidelberg.

[11] Z.-q. Du, Q.-j. Xu, J. Zhang, and M. Li, "Design and analysis of dynamic s-box based on feistel," in Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), 2015 IEEE, pp. 590–594, IEEE, 2015.