Multimedia Tools and Applications (2022) 81:18875–18895
https://doi.org/10.1007/s11042-022-12742-1

A unified improvement of the AES algorithm

Yong Zhang [1] · Aiguo Chen [1] · Bin Chen [1]

Received: 28 January 2021 / Revised: 22 April 2021 / Accepted: 21 February 2022 /
Published online: 9 March 2022

Abstract
Symmetric cryptography is widely used in information exchange and storage. The advanced encryption standard (AES) is one of the most important symmetric ciphers. Based on the AES algorithm, this paper designs a new symmetric cipher, called unified algorithm. In the unified algorithm, the secret key is 256-bit long and the plaintext and cipher-text are both 128-bit long. The unified algorithm is composed of the modules and their improved versions of AES and a sequence-flip module, and has the same encryption and decryption processes. The simulation results show that the unified algorithm has the same processing speed as AES, and its single-thread running speed in Mathematica can reach up to 1.6570Mbps. Meanwhile, the unified algorithm has the same encryption intensity as AES, and its relative errors of system sensitivity index are less than 0.5%. The unified algorithm can be applied to various secure communication occasions to replace the AES algorithm. In addition, due to the encryption and decryption sharing the same module, the unified algorithm can save the hardware resources and simplify the secure communication protocol effectively.

Keywords Advanced encryption standard (AES) · Unified algorithm · Key expansion · Symmetric cipher · System sensitivity

1 Introduction

Compared with the public key cryptography, symmetric key cryptography has the advantages of fast encryption speed and short key [13, 24]. At present, the widely used symmetric ciphers are divided into two types, stream cipher and block cipher. The representative of stream cipher is RC4 [11], which produces cipher-text by performing exclusive-OR operation between the plaintext and key streams, and is mainly used in computer network communication. The block ciphers have DES [8], AES [4], IDEA [2], BlowFish [23], SM4 [26] and etc. [16, 17], which

✉ Yong Zhang
zhangyong@jxufe.edu.cn

[1] School of Software & Internet-of-Things Engineering, Jiangxi University of Finance & Economics, Nanchang 330013, People's Republic of China

are commonly applied to information exchange and storage. AES is the most frequently used symmetric cipher [14, 18, 25] due to its high security level and high-speed implementation in both hardware and software.

Before AES was confirmed as the text encryption standard by NIST (National Institute of Standards and Technology) in 2001, DES was the most popular block cipher. DES and BlowFish are both based on the Feistel network. A remarkable feature of the Feistel network is to make the encryption module and decryption module of DES exactly the same [10, 22]. Thus, DES can save half of the resources when implemented in hardware or software. However, the key arrangement algorithms of DES are different in the processes of encryption and decryption. So, a complete DES algorithm needs an encryption or decryption module and two reciprocal key arrangement modules. Furthermore, the key length of DES is only 56 bits, which cannot resist the exhaustive attack of current high-performance computers, and DES has been replaced by TDEA and AES [9, 20].

The AES algorithm is a block cipher proposed by J. Daemen and V. Rijmen [7]. The block size is 128 bits, and the key length is 128, 192 or 256 bits. As the text encryption standard of NIST, AES is widely used in Internet security standards, such as IPSec, TLS, and SSH, and WiFi wireless security standard of IEEE 802.11i to encrypt the sensitive and important information [5, 19, 29, 32]. The AES algorithm is based on 8-bit byte and 32-bit word processing, which is suitable for the microprocessor and ASIC chip hardware implementation. Meanwhile, it also can be implemented in computer software based on the look-up table method at high speed [15, 28]. Practice shows that AES can be used to encrypt documents of various secret levels.

AES was originally designed to encrypt text data, but now there are a variety of extended algorithms based on AES for encrypting a large quantity of data or image information [1, 6, 12]. In [6], the elliptic curve cryptography (ECC) is combined with AES to encrypt color images. ECC provides the key distribution, and AES is used to encrypt each channel of color images. In [1], an image encryption scheme based on AES is proposed. The pseudo-random sequence generated by chaotic system is used to replace the elements of S-box of AES for byte substitution operations, and then the chaotic state sequence is employed to scramble the outcome to enhance the security performance of cipher images. In [12], an image information security scheme combining watermark and compression is proposed. AES in cipher-chain mode (CBC) is used to encrypt the watermarked and compressed image. In addition, the substitution box (S-box) of AES, as the main nonlinear component, is frequently used in various image cryptosystems [3, 21, 34]. These systems realize the confusion of image information with the help of S-box.

Inspired by DES and AES, image cryptography based on chaotic system has been developed rapidly [27, 35]. In 2017, Zhang proposed an image encryption system with one module shared by encryption and decryption [30]. Due to the plaintext-related confusion, the image encryption system has strong plaintext sensitivity. However, similar to DES, this system used the reciprocal key arrangement in the encryption and decryption processes. Then, Zhang et al. improved the algorithm in [30] and proposed an image cryptosystem in which the encryption process and decryption process, including the key expansion module, are completely shared [33]. After that, on the basis of [33], Zhang proposed a fast image cryptosystem with one shared module based on the lifting transformation [31]. This system only used diffusion approach, which further improved the speed of image encryption.

Although AES was proposed as an alternative to DES, AES did not possess the advantage that the encryption module and decryption module of DES shared the same algorithm. The

structures of encryption module and decryption module of AES are different, and they are reverse to each other. Inspired by the sharing method of encryption and decryption of DES and the sharing algorithm in image cryptography, this paper proposes a new fast symmetric cryptography based on AES. The main contributions of this paper are as follows: (i) propose a new sharing algorithm of encryption and decryption based on the modules of AES, called unified algorithm; (ii) generalize the "ShiftRows" module of AES; (iii) present the cipher-chain mode of unified algorithm to encrypt a large amount of data. Furthermore, this paper compares and analyzes the encryption speed and encryption intensity of unified algorithm and AES in detail. The remainder of this paper is arranged as follows: Section 2 reviews the AES algorithm; Section 3 describes the unified algorithm and its implementation; Section 4 compares the processing speed and security strength of unified algorithm and AES; Section 5 summarizes the full text.

2 The algorithm of AES

AES, also known as the Rijndael algorithm, has the structure as shown in Fig. 1. In AES, the secret key k is 128, 192 or 256 bits, and the corresponding rounds N_r are 10, 12 or 14. For the 256-bit AES, NIST pointed out that it can be used for data encryption in the case of top secret. To save space, this paper only discusses the AES algorithm with the key length of 256 bits, that is, $N_r = 14$ and k is 256-bit long in Fig. 1.

In Fig. 1, a 256-bit key k is the input for "Key expansion" module to be transformed into 15 pieces of sub-keys $\{k_0, k_1, \ldots, k_{14}\}$, and each sub-key is 128-bit long. As shown in Fig. 1a, the encryption process of AES performs the following steps: (i) Carry out the bitwise exclusive-or operation between the plaintext x and the sub-key k_0; (ii) For the outcome of (i), perform the "SubBytes", "ShiftRows", "Mixcolumns" and "bitwise exclusive-or with sub-key k_i" operations in turn circularly for N_r-1 times, where i is the number of round; (iii) For the result of (ii), perform the "SubBytes", "ShiftRows" and "bitwise exclusive-or with sub-key k_{Nr}" operations in turn to obtain the cipher-text y. The decryption process of AES is the inverse of the above encryption, as shown in Fig. 1b, where, "ShiftRows^{-1}" is the inverse of "ShiftRows".

2.1 Key expansion module

In Fig. 1, the key expansion used in encryption process and decryption process of AES is the same, but the order of sub-keys is reverse. The key expansion module is shown in Fig. 2, where, "$\oplus$" represents bitwise exclusive-or operation. The key expansion module first divides the key k into 8 pieces of 32-bit words, denoted by $\{w_i\}$, $i = 0,1,\ldots,7$. Then, according to the algorithm shown in Fig. 2, this module produces the words $\{w_i\}$, $i = 8,9,\ldots,59$. The sub-keys in Fig. 1 are $k_i = \{w_{4i}, w_{4i+1}, w_{4i+2}, w_{4i+3}\}$, $i = 0,1,\ldots,14$. The functions g and h in Fig. 2 are shown in Fig. 3.

In Fig. 3, "S" stands for the substitution box (S-box), as shown in Fig. 4. For the function g in Fig. 3a, a 32-bit word is firstly divided into four 8-bit bytes. Secondly, each byte looks up the S-box in Fig. 4 to get a new 8-bit byte. Thirdly, the byte derived from v_1 and S-box does exclusive-or operation with rc_i. Finally, the resulted four bytes are combined by the index order {2,3,4,1} into a new 32-bit word. With respect to the look-up table operation of some byte v, the upper four bits of v are regarded as row number *row* of S-box in Fig. 4, and the lower four bits of v as column number *col* of S-box, then S-box(*row*,*col*) is the value of look-

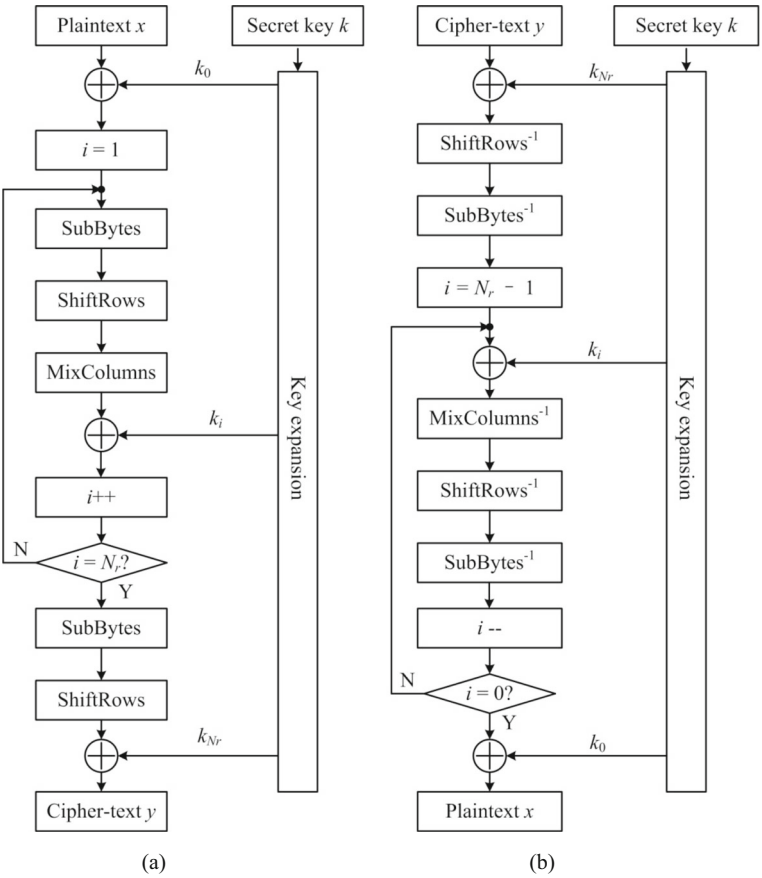


Fig. 1 The algorithm of AES. **a** Encryption algorithm; **b** Decryption algorithm

up S-box by v. For the function h in Fig. 3b, a 32-bit word is the input, and its four bytes separately look up S-box to get the new bytes which are combined into the resultant word in the original order.

2.2 The processing algorithm of encryption round

Each round of processing algorithm in AES encryption includes four parts, "SubBytes", "ShiftRows", "MixColumns" and "bitwise exclusive-or operation with the round sub-key", as shown in Fig. 5. In each round, the input is a 128-bit bit-sequence, which is divided into 16 bytes, denoted by $\{A_j\}, j = 0,1,\ldots,15$. Then, each byte looks up S-box to get a new byte, and the outcome is denoted by $\{B_j\}, j = 0,1,\ldots,15$. The above is called "SubBytes". Next, the "ShiftRows" rearranges $\{B_j\}, j = 0,1,\ldots,15$ into a new sequence $\{B_k\}, k = 0,1,\ldots,15$, satisfying $j = 5\ k \bmod 16$. After that, every four bytes of the sequence $\{B_k\}$ are multiplied by a matrix M_1 in the "MixColumns" module. The matrix multiplication and addition operations are based on GF(2^8) field with the irreducible polynomial $P(x) = x^8 + x^4 + x^3 + x + 1$. The matrix M_1 is as follows:

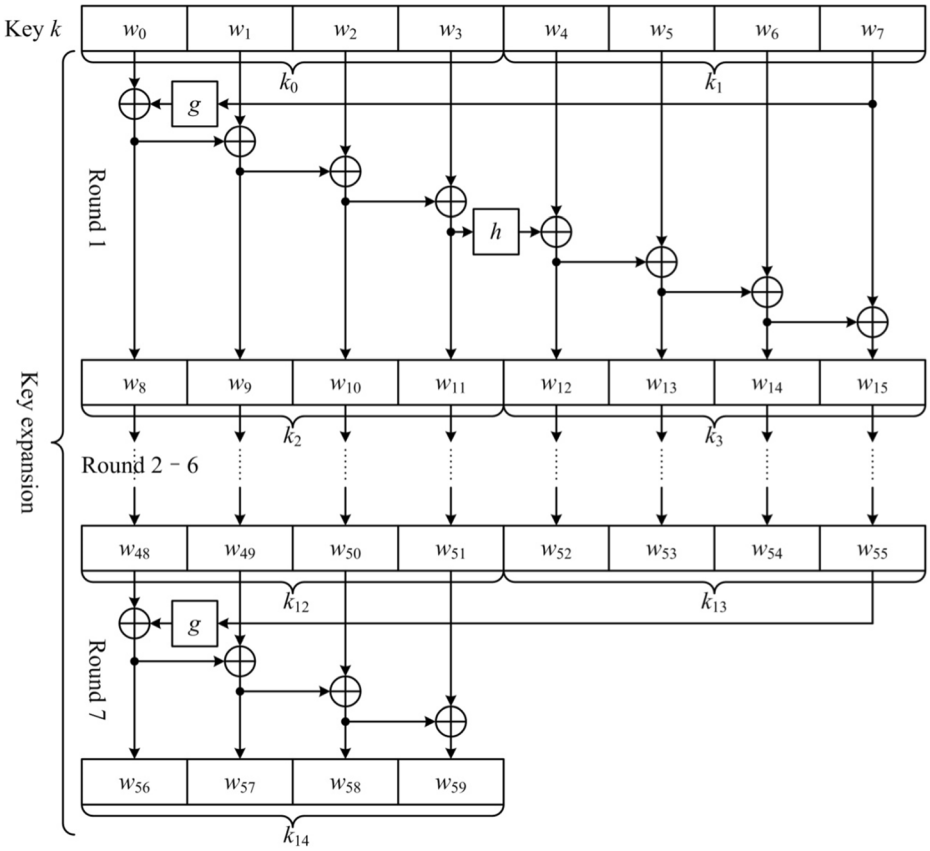


Fig. 2 The key expansion module

$$M_1 = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \tag{1}$$

After the "MixColumns" operation, the resultant 16 bytes are combined into a new 128-bit bit-sequence, which is carried out the exclusive-or operation with the round sub-key k_i to get the output of this round. Here, k_i is the sub-key of the ith round.

2.3 The processing algorithm of decryption round

The round processing in AES decryption, as shown in Fig. 6, is the inverse of round processing in AES encryption, including four parts, the "exclusive-or operation with the round sub-key k_i", "MixColumns^{-1}", "ShiftRows^{-1}" and "SubBytes^{-1}", where, "Module^{-1}" means the inverse of the "Module". In the i-th round, the input is a 128-bit bit-sequence, which is executed the exclusive-or operation with the round sub-key k_i, to obtain a new bit-sequence. The new bit-sequence is divided into 16 bytes, denoted by $\{C_j\}, j = 0,1,\ldots,15$. Then, in the

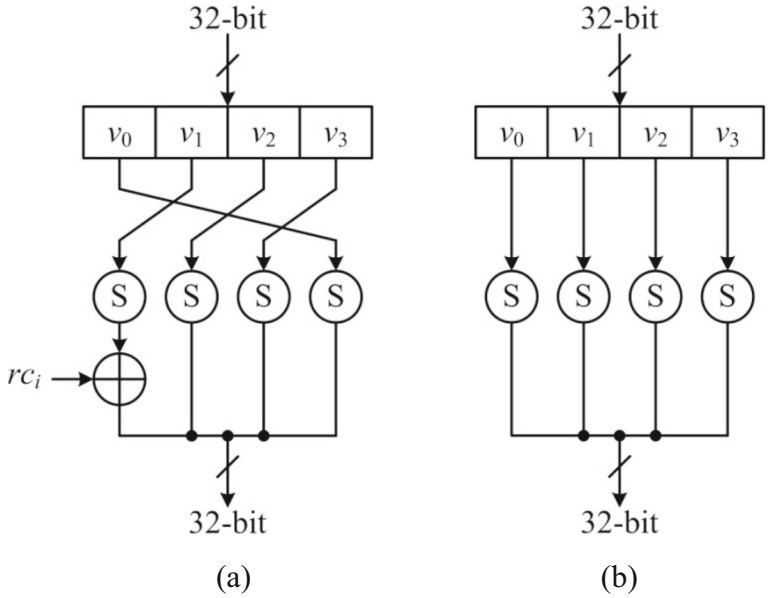


Fig. 3 The structures of g and h. **a** The function g in the round i, where, $rc_i = 1 << (i\text{-}1)$, $i = 1,2,\ldots,7$. **b** The function h

"MixColumns^{-1}", every four bytes of $\{C_j\}$ are multiplied by a matrix M_2 to fulfill the column mixture. The matrix M_2 is as follows:

$$M_2 = \begin{bmatrix} 14 & 11 & 13 & 9 \\ 9 & 14 & 11 & 13 \\ 13 & 9 & 14 & 11 \\ 11 & 13 & 9 & 14 \end{bmatrix} \tag{2}$$

		col															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
row	0	99	124	119	123	242	107	111	197	48	1	103	43	254	215	171	118
	1	202	130	201	125	250	89	71	240	173	212	162	175	156	164	114	192
	2	183	253	147	38	54	63	247	204	52	165	229	241	113	216	49	21
	3	4	199	35	195	24	150	5	154	7	18	128	226	235	39	178	117
	4	9	131	44	26	27	110	90	160	82	59	214	179	41	227	47	132
	5	83	209	0	237	32	252	177	91	106	203	190	57	74	76	88	207
	6	208	239	170	251	67	77	51	133	69	249	2	127	80	60	159	168
	7	81	163	64	143	146	157	56	245	188	182	218	33	16	255	243	210
	8	205	12	19	236	95	151	68	23	196	167	126	61	100	93	25	115
	9	96	129	79	220	34	42	144	136	70	238	184	20	222	94	11	219
	10	224	50	58	10	73	6	36	92	194	211	172	98	145	149	228	121
	11	231	200	55	109	141	213	78	169	108	86	244	234	101	122	174	8
	12	186	120	37	46	28	166	180	198	232	221	116	31	75	189	139	138
	13	112	62	181	102	72	3	246	14	97	53	87	185	134	193	29	158
	14	225	248	152	17	105	217	142	148	155	30	135	233	206	85	40	223
	15	140	161	137	13	191	230	66	104	65	153	45	15	176	84	187	22

Fig. 4 S-box

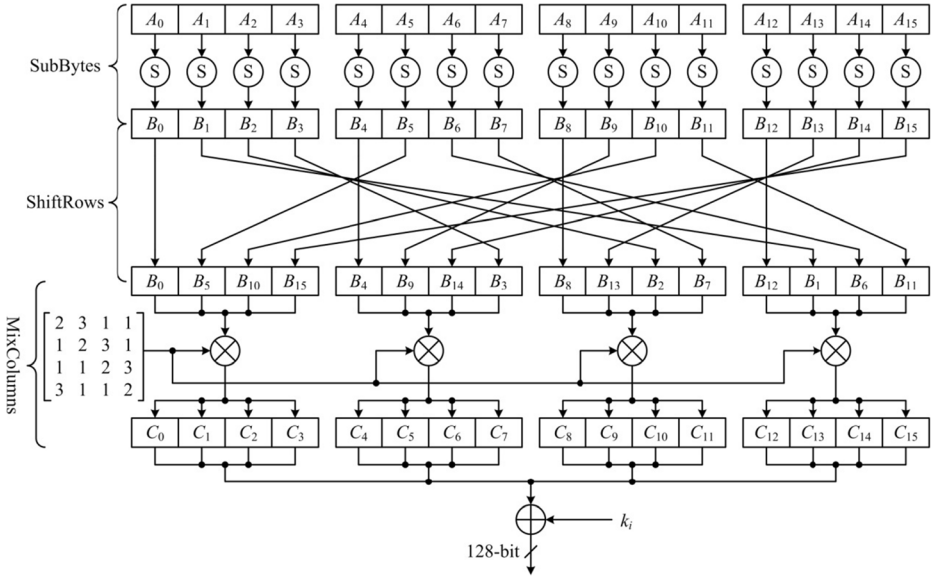


Fig. 5 The round processing algorithm for the encryption module of AES

The output of "MixColumns^{-1}" is denoted by $\{B_j\}, j = 0,1,\ldots,15$. Then, the "ShiftRows^{-1}" module rearranges $\{B_j\}, j = 0,1,\ldots,15$ into $\{B_k\}, k = 0,1,\ldots,15$, satisfying $j = 13\ k \bmod 16$. Next, the "SubBytes^{-1}" module carries out look-up table operation on the sequence $\{B_k\}$ to get a new sequence, denoted by $\{A_k\}, k = 0,1,\ldots,15$. The used look-up table is the inverse of S-box, as shown in Fig. 7.

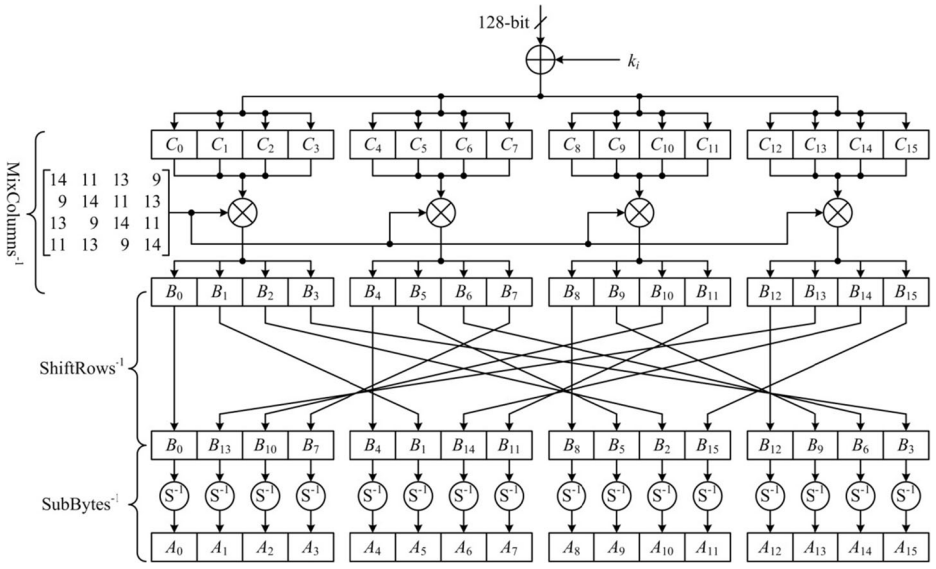


Fig. 6 The round processing algorithm for the decryption module of AES

		col															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
row	0	82	9	106	213	48	54	165	56	191	64	163	158	129	243	215	251
	1	124	227	57	130	155	47	255	135	52	142	67	68	196	222	233	203
	2	84	123	148	50	166	194	35	61	238	76	149	11	66	250	195	78
	3	8	46	161	102	40	217	36	178	118	91	162	73	109	139	209	37
	4	114	248	246	100	134	104	152	22	212	164	92	204	93	101	182	146
	5	108	112	72	80	253	237	185	218	94	21	70	87	167	141	157	132
	6	144	216	171	0	140	188	211	10	247	228	88	5	184	179	69	6
	7	208	44	30	143	202	63	15	2	193	175	189	3	1	19	138	107
	8	58	145	17	65	79	103	220	234	151	242	207	206	240	180	230	115
	9	150	172	116	34	231	173	53	133	226	249	55	232	28	117	223	110
	10	71	241	26	113	29	41	197	137	111	183	98	14	170	24	190	27
	11	252	86	62	75	198	210	121	32	154	219	192	254	120	205	90	244
	12	31	221	168	51	136	7	199	49	177	18	16	89	39	128	236	95
	13	96	81	127	169	25	181	74	13	45	229	122	159	147	201	156	239
	14	160	224	59	77	174	42	245	176	200	235	187	60	131	83	153	97
	15	23	43	4	126	186	119	214	38	225	105	20	99	85	33	12	125

Fig. 7 The inverse of S-box

3 Unified algorithm

On the basis of AES, the proposed unified algorithm, as shown in Fig. 8, is the cryptographic algorithm with the same encryption process and decryption process. The modules of "SubBytes", "SubBytes^{-1}", "MixColumns" and "MixColumns^{-1}" in Fig. 8 are the same as the modules with the same name in Fig. 1. However, the modules of "ShiftRows" and "ShiftRows^{-1}" in Fig. 8 are modified based on the modules with the same name in Fig. 1. The following subsections will introduce the implementation of unified algorithm in detail.

Here, only the system with 256-bit key is considered, and in Fig. 8, $N_r = 14$.

3.1 Modified key expansion module

The modified key expansion module in Fig. 8 is based on the key expansion module in Fig. 2. First, generate the sub-keys $\{k_i\}$, $i = 0,1,\ldots,N_r$ by the key expansion in Fig. 2. Then, use these sub-keys to produce the sub-keys $\{key_i\}$, $i = 0,1,\ldots, N_r/2$ in Fig. 8, such that $key_0 = k_0$, $key_i = k_{2i-1}$ XOR k_{2i}, $i = 1,2,\ldots,N_r/2$, where, "XOR" means the exclusive-or operation. In this way, the modified key expansion module can be executed in parallel with the encryption/decryption process of unified algorithm.

3.2 Modified "ShiftRows" and its inverse algorithm

In the standard AES, "ShiftRows" realizes the cyclic shift of byte sequence. Denote the sequences before and after the "ShiftRows" module as $\{B_j\}$, $j = 0,1,\ldots,15$ and $\{D_k\}$, $k = 0,1,\ldots,15$, respectively. Then, the operation of "ShiftRows" is to put $D_k = B_{5k \bmod 16}$. And the operation of "ShiftRows^{-1}" is to put $B_j = D_{13j \bmod 16}$. Obviously, "ShiftRows" and "ShiftRows^{-1}" are reciprocal.

For any integer n and $v \in \{0, 1, \ldots, 15\}$, one has $(16n + 1)v \bmod 16 = v$. Take the value of n as 2,3,4 and 5 to obtain the values of $16n + 1$ and their factors (such that each factor is less than 16), as shown in Table 1.

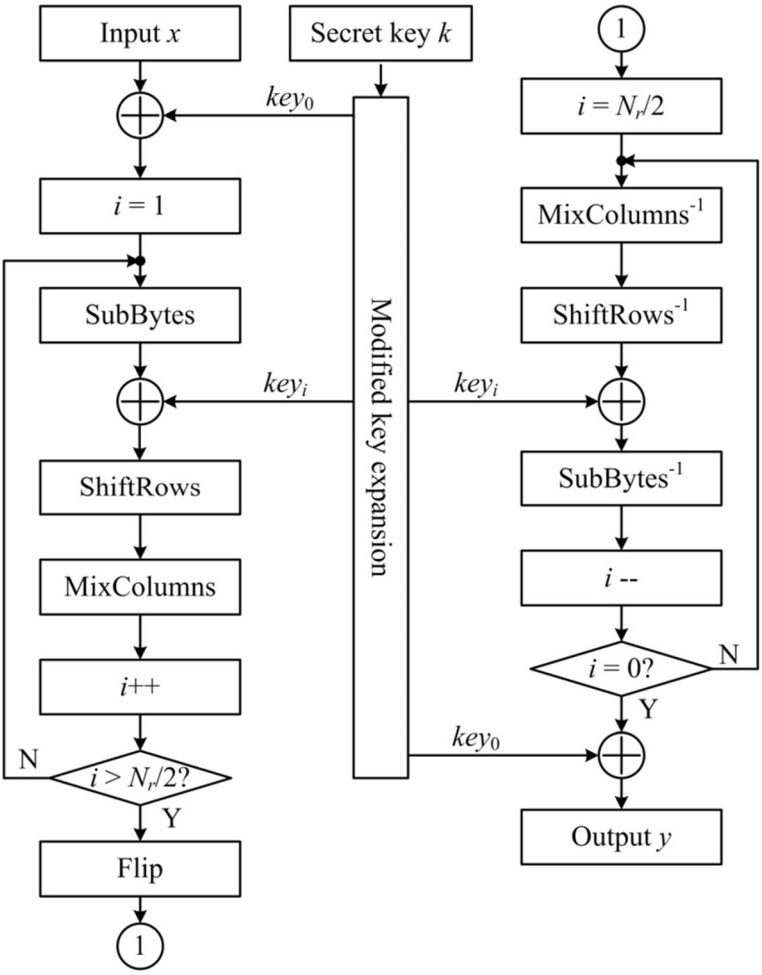


Fig. 8 The unified algorithm based on AES

In Table 1, when $n = 4$, the two factors are 5 and 13, which are used as the shift values of "ShiftRows" and "ShiftRows^{-1}" in AES, respectively. In the unified algorithm shown in Fig. 8, the shift values of the modified "ShiftRows" and "ShiftRows^{-1}" modules are listed in Table 1. The round "i" in the last column of Table 1 represents "i" in Fig. 8. Table 1 shows that (i) in the two rounds of $i = 1$ and 7 in Fig. 8, the "ShiftRows" and "ShiftRows^{-1}" modules will use 3 and 11 as the shift values, respectively; (ii) in the two rounds of $i = 3$ and 5, the two modules will both use 7; (iii) in the two rounds of $i = 2$ and 6, the two modules will use 5 and 13, respectively (i.e. the case of AES); (iv) in the round $i = 4$, the modules will both use 9.

3.3 The implementation process of unified algorithm

For the unified algorithm, the input is a secret key k and a 128-bit bit-sequence x, and the output is another 128-bit bit-sequence y. If the input bit-sequence x is a plaintext, the output y is its cipher-text. And if the input x is a cipher-text, the output y will be its recovered plaintext.

Table 1 The value of n and the factorization of $16n + 1$

n	$16n+1$	Factors	Round i
2	33	3×11	1, 7
3	49	7×7	3, 5
4	65	5×13	2, 6
5	81	9×9	4

In Fig. 8, the secret key k is fed to the modified key expansion to obtain the sub-keys $\{key_i\}$, $i = 0,1,\ldots,N_r/2$. Then, the input x is transformed into the output y using the sub-keys $\{key_i\}$ with the following steps:

Step 1. Perform the exclusive-or operation between x and key_0 to get a sequence, denoted by tp_0.
Step 2. Repeat the following operation $N_r/2$ times. For the i-th round, the input is tp_{i-1} and the output is tp_i, $i = 1,2,\ldots,N_r/2$.

For the i-th round, tp_{i-1} is fed to the "SubBytes" module, and then the outcome is executed the exclusive-or operation with the sub-key key_i. Next, the resultant sequence is handled by the "ShiftRows" and "MixColumns" modules in turn to produce the output sequence tp_i.

Note that the shift values in the "ShiftRows" module are 3, 5, 7, 9, 7, 5 and 3 in turn in this step.

Step 3. Flip the sequence $tp_{Nr/2}$ to get a new sequence, denoted by $tq_{Nr/2}$, i.e. tq_7. When this step is implemented on the computer, the flip operation is in bytes; when this step is implemented on the microprocessor or ASIC chip, the flip operation is in bits. To save space, this paper only considers the sequence flip operation in bytes.
Step 4. Repeat the following operation $N_r/2$ times. For the i-th round, the input is tq_i and the output is tq_{i-1}, $i = N_r/2, N_r/2–1,\ldots,1$.

For the i-th round, tq_i is firstly handled by the "MixColumns^{-1}" and "ShiftRows^{-1}" modules in turn, and then the outcome is executed the exclusive-or operation with the sub-key key_i. Next, the resultant sequence is fed to the "SubBytes^{-1}" module to produce the output sequence tq_{i-1}.

Note that the shift values of the "ShiftRows^{-1}" module in this step are 11, 13, 7, 9, 7, 13 and 11 in turn.

Step 5. Do the exclusive-or operation between tq_0 and key_0 to get the output y.

The above algorithm is inverse to itself, that is to say, the encryption process is identical to the decryption process. Thus, it is called unified algorithm.

Compared with the standard AES, the unified algorithm has the following characteristics:

(i) Due to the encryption and decryption share the same process, the unified algorithm has the same encryption speed and decryption speed in theory, while the encryption speed and decryption speed of AES are slightly different.

(ii) The unified algorithm uses both the encryption module and decryption module of AES.
(iii) The key expansion module of unified algorithm is mainly based on the key expansion module of AES, which can be paralleled with the encryption/decryption process. The processing speed of unified algorithm is equivalent to that of AES.
(iv) The unified algorithm has one more flip module than AES, and the flip module can be easily implemented by the computer software and ASIC hardware.
(v) The shift value of "ShiftRows" of each round in the unified algorithm is related to the number of round, while the shift value in AES is fixed. Thus, the unified algorithm increases the way of diffusion.
(vi) The unified algorithm includes seven rounds of encryption modules and seven rounds of decryption modules, i.e. with a total of 14 rounds of AES modules. The overall load is equivalent to that of AES.
(vii) The unified algorithm saves about half of the resources and simplifies the secure communication protocol by providing the same encryption and decryption processes.

The following section will further verify by simulation test that the unified algorithm based on AES has the same processing speed and security intensity as AES. To save space, only the cases of unified algorithm and AES both with the 256-bit key are compared and analyzed.

4 Simulation and comparative analysis

The computer is equipped with AMD Ryzen 93,950X@3.50GHz CPU, 64GB DDR4@3200 MHz memory, and 64-bit Windows 10 Home. The simulation is based on Mathematica 12.1.1 with the Wolfram language and its "Compile" modules under MinGW64 compiler. The objects of simulation include processing speed comparison analysis and system sensitivity comparison analysis.

4.1 Speed comparison analysis

The computer programs of the unified algorithm and AES are realized in the Wolfram language with MinGW64 compiler. Note that the programs are designed according to the modules in Figs. 1 and 8, without using the look-up table method to optimize the "ShiftRows" and "MixColumns" and their reverse modules so as to fairly compare the processing speed of the two algorithms.

For AES, the encryption speed and decryption speed are measured as follows:

(i) Randomly generate a key k of length 256 bits and a plaintext x of length 128 bits. Then, use AES to encrypt x with k to get the corresponding cipher-text y. Denote the encryption time consumed as t_1. Next, use AES to decrypt y with k to recover the plaintext x, and denote the decryption time as t_2.
(ii) (ii) Repeat the above process 10,000 times, and calculate the average encryption time and decryption time, separately denoted by t_e and t_d both with the unit μs. Then, label the encryption speed $v_e = 128/t_e$ Mbps and the decryption speed $v_d = 128/t_d$ Mbps.

For the unified algorithm, the encryption process is identical to its decryption process, so only one processing speed needs to be recorded as follows:

(i) Randomly produce a key k of length 256 bits and a plaintext x of length 128 bits. Then, use the unified algorithm to encrypt x with k to get the corresponding cipher-text y. Denote its encryption time as t_1. Next, use the unified algorithm to decrypt y with k to recover the plaintext x, and denote its decryption time as t_2. Now, calculate the average time of $t_a = (t_1 + t_2)/2$.

(ii) Repeat the above process 5000 times, and calculate the average value of all the t_a, denoted by t_u with the unit μs. Then, label the processing speed $v_u = 128/t_u$ Mbps.

According to the above method, the processing speeds of the unified algorithm and AES are calculated and the results are listed in Table 2.

From Table 2, it can be seen that (i) the encryption speed and decryption speed of AES are 1.7750Mbps and 1.7467Mbps, respectively; (ii) the processing speed of unified algorithm is 1.6570Mbps, which is about 6% slower than that of AES. Thus, the processing speeds of the unified algorithm and AES are at the same level.

4.2 Sensitivity comparison analysis

AES is a mature symmetric cryptography with good security intensity. This subsection will compare the security intensity of unified algorithm and AES, mainly in four aspects, encryption-key sensitivity, decryption-key sensitivity, plaintext sensitivity and cipher-text sensitivity. The comparative analysis will use the following difference index *diff*.

For two bit-sequences A and B of the same length n, the difference index *diff* is defined as

$$diff = \frac{Count(A \oplus B, 1)}{n} \times 100\% \tag{3}$$

where "⊕" is the bitwise exclusive-or operation, and $Count(x,1)$ returns the number of bit 1 in the bit-sequence x. For two random bit-sequences, the theoretical value of their *diff* is 50%; for a given bit-sequence and a random bit-sequence, the theoretical value of their *diff* is also 50%.

4.2.1 Comparative analysis of encryption-key sensitivity

Any cryptosystem should avoid using special form of secret key. Here, five random keys and five random plaintexts are generated for analyzing the sensitivity of encryption key. The five keys are all of 256-bit length and in hexadecimal form, as follows:

k_1 = {C564ECD2B638ACA4DB22EEA32BB4396981043F07145642DBE43C
767B195D5D6}.
k_2 = {170E37334A355B01567F523D2B67A06A3CAB0065DD65D5BF4
42232C96BD327B0}.

Table 2 The processing speeds of the unified algorithm and AES

Item	AES		Unified algorithm
	Encryption	Decryption	
Time (μs)	72.112	73.281	77.246
Speed (Mbps)	1.7750	1.7467	1.6570

k_3 = {886CECAD20D83EC388311FE614A7AA29A5DA775709A
041FE1370942ADD347045}.
k_4 = {5159B689F65710461AA84D9F0E96CF1D706964C871A3F9270888
7B0EE7D27D4F}.
k_5 = {D814EF865655F589C57292BBA1C922E8A41E64731
3F142F122BECEA6FAD16782}.

And the five plaintexts are all of128-bit length and in hexadecimal form, as follows:

x_1 = {7C1D93571A35B4366185CEFF22A8164F}.
x_2 = {C0077BE6B566007275399E63F9F75E12}.
x_3 = {37CB1D44997D112934A4BB86B897C127}.
x_4 = {F9D6D827BF690C13A79F929C4174DA04}.
x_5 = {643DC545A720EB59AADEACD514C760A9}.

The results of encryption-key sensitivity test using the above keys and plaintexts are listed in Table 3. In Table 3, "$k_i(j)$" represents the key k_i with only its j-th bit changed, where, i = 1,2,…,5, and j can take one of {0,1,…,255}, and all the cipher-texts are in hexadecimal form.

In Table 3, in each test, change one-bit of the given key to get a new key. Then, use AES or the unified algorithm to encrypt a plaintext with the two keys to get two corresponding cipher-texts. Next, calculate the *diff* between the two cipher-texts. It can be seen from Table 3 that the calculated values of *diff* of AES and the unified algorithm fluctuate about the theoretical value 50% with the maximum amplitude of 8%.

In order to make the analysis results have common significance, 10,000 trails are carried out for comparative analysis. In each trial, randomly generate a 256-bit key k_1 and a plaintext x at first. Secondly, randomly change one bit of k_1 to get a new key, denoted by k_2. Thirdly, Use AES with k_1 and k_2 to encrypt x to get two cipher-texts, denoted by y_1 and y_2, respectively. Fourthly, calculate the index *diff* between y_1 and y_2. Fifthly, repeat the above steps 10,000 times to record the maximum, average and minimum values of *diff*. Finally, use the unified algorithm to replace AES and then repeat the above process to obtain the maximum, average and minimum values of *diff* for the unified algorithm. The test results are listed in Table 4.

The last column in Table 4 shows the relative error between the average value of *diff* and the theoretical value 50%. As can be seen from Table 4, the calculated values of *diff* of AES and the unified algorithm are similar, and the relative errors are both less than 0.5%, indicating that the unified algorithm has the same strong encryption-key sensitivity as AES.

4.2.2 Comparative analysis of decryption-key sensitivity

The steps of decryption-key sensitivity analysis are as follows:

(i) Randomly generate a 256-bit key k_1 and a plaintext x_1.
(ii) Use AES with k_1 to encrypt x_1 to get the corresponding cipher-text y.
(iii) Randomly change k_1 by one-bit to get a new key, denoted by k_2.
(iv) Use AES with k_2 to decrypt y to get the recovered sequence, denoted by x_2.
(v) Calculate the index *diff* between x_1 and x_2.
(vi) Repeat the above steps 10,000 times to record the maximum, average and minimum values of *diff*.

Table 3 Results of typical examples for encryption-key sensitivity comparison analysis

No.	Key	Plaintext	AES		Unified algorithm	
			Cipher-text	*diff*	Cipher-text	*diff*
1	k_1	x_1	69C9B8B110E8FE17 865A5D19317E8A2B	47.6563%	389F98AAEE706AFE E71F896A57DB8C04	56.2500%
	$k_1(0)$		DACD8627B0FDE867 A1A8FF504A208755		7A69BF47A1EDCC80 4280A8F9210DD766	
2	k_2	x_2	23A4BF36D46C23F4 52BA01EEC9298500	42.1875%	E7C6B17DE31DB77A F1DC8D2E804E6B16	53.9063%
	$k_2(27)$		F1AD9800B407126E 5CFE92D629AEA1F1		B6B0514C906AB97F DE62E6195928E1A8	
3	k_3	x_3	6D706DA4BEE2734A 5FB096238EEEC1BB	56.2500%	94B47DF9767B9665 436E13BD0DCA8786	53.1250%
	$k_3(64)$		30FE92384B2FC6A7 12B0E5AFBBBDA767		6CCE9100DFC7E212 3BF7008B07FE13DB	
4	k_4	x_4	71944D312BDE1CF2 BA1CACD166F66FCA	47.6563%	CDC71D72538583ED BF3C925100A9F350	46.8750%
	$k_4(190)$		F76767891C642A33 988E6CCFA01CC729		06518A760A01F84F 847CFEE6869CCBB8	
5	k_5	x_5	16C94700D996CDDE E70BEE062BC4D7C4	52.3438%	2E858D4140914D5B 1301064CB35A6317	50.7813%
	$k_5(255)$		EE15036C935DBAF9 39001368B9D99DE6		769E01F20B5DF238 74020DE9AFA70501	

(vii) Repeat the above processing with the unified algorithm instead of AES to obtain the maximum, average and minimum values of *diff* for the unified algorithm.

The test results are listed in Table 5.

The last column in Table 5 shows the relative error between the average value of *diff* and the theoretical value 50%. From Table 5, it can be seen that the calculated values of *diff* of the unified algorithm are close to those of AES, and the relative errors are both less than 0.5%. Thus, like AES the unified algorithm has strong decryption-key sensitivity.

4.2.3 Comparative analysis of plaintext sensitivity

To intuitively compare the plaintext sensitivity between AES and the unified algorithm, the following five plaintexts all of 128 bits in hexadecimal form are randomly selected,

x_1 = {CAEE7121E4F10DD7E376261765C23994},
x_2 = {6EE4E32EF9E98DCC6D6E385C1F8F72A5},
x_3 = {ACC5070D9A0A9B232C4EDA1960B36E28},

Table 4 The calculation results of *diff* in encryption-key sensitivity comparison analysis

Algorithm	*diff*			Relative error of the average value
	Maximum	Minimum	Average	
AES	67.1875%	32.0313%	49.9891%	0.0218%
Unified Algorithm	65.6250%	34.3750%	50.0047%	0.0094%

Table 5 The calculation results of *diff* in decryption-key sensitivity comparison analysis

Algorithm	diff			Relative error of the average value
	Maximum	Minimum	Average	
AES	66.4063%	33.5938%	50.0627%	0.1254%
Unified algorithm	66.4063%	32.8125%	49.9801%	0.0398%

x_4 = {435B33B71C9F6C4B3AF6B40D3C54F854},
x_5 = {19D33CBE05F1E36EB187728CFE481EE8}.

And then the following five keys all of 256 bits in hexadecimal form are randomly selected,

k_1 = {E41A3DC6AF2B5A0A25797B295E936614C4B716F00EE027867F969A
C5F0613DDC},
k_2 = {CFFA75E2CD9692312F3331E7E113444534F11C42B671C2A9ACD2946
613B6274C},
k_3 = {748BE7776617B40F5939D3C307BD6EF4960650E1E744830601CD592D
73FAE5D7},
k_4 = {151608C7AFA085DDB5C253BCFE8AFE774F71715B8B5EBF825967998
CF0E0BB75},
k_5 = {10128FAA25793B10734C5911014429F6B98DC4A2C920C7DBEC6C
89505BA8613}.

Next, the results of plaintext sensitivity analysis with the help of the above keys and plaintexts are listed in Table 6. In Table 6, "$x_i(j)$" represents the plaintext x_i with only its j-th bit changed, where, i = 1,2,…,5, and j can take one of {0,1,…,127}, and all the cipher-texts are in hexadecimal form.

As can be seen from Table 6, the comparison results of five examples show that the calculated values of index *diff* of the unified algorithm are close to those of AES, both fluctuating about the theoretical value 50% with the maximum amplitude of 8%.

To make the comparison results have universal significance, 10,000 trials are carried out for the comparative analysis. In each trial, randomly generate a secret key k at first. Then, randomly generate a plaintext x_1. Next, randomly change x_1 by one bit to get a new plaintext, denoted by x_2. Now, use AES with k to encrypt x_1 and x_2 to get their corresponding cipher-texts, denoted by y_1 and y_2, respectively. After that, calculate the index *diff* between y_1 and y_2. Repeat the above steps 10,000 times to obtain the maximum, average and minimum values of *diff*. Finally, repeat the above process using the unified algorithm instead of AES to get the maximum, average and minimum values of *diff* for the unified algorithm. The test results are listed in Table 7.

The last column in Table 7 shows the relative error between the average value of *diff* and the theoretical value 50%. As can be seen from Table 7, the calculated values of *diff* of the unified algorithm are close to those of AES, and the relative errors are both less than 0.5%. Thus, the unified algorithm has the same strong plaintext sensitivity as AES.

Table 6 Results of typical examples for plaintext sensitivity comparison analysis

No.	Plaintext	Key	AES		Unified algorithm	
			Cipher-text	*diff*	Cipher-text	*diff*
1	x_1	k_1	98DA0DCD097AEAD9 F2B7F7D19F3E930C	46.0938%	3D1DF03F51EBBDF2 F5E9B46E9CA23018	54.6875%
	$x_1(0)$		B4B49480410D4F7D DE275DCCA969420A		43D347BECAD9AA55 C8A78D8A9B7C0A74	
2	x_2	k_2	F4E2840E0AF391B2 81C7BE0E1BC6415F	48.4375%	0831A65FBD5A6BD1 EAB13261CD093944	52.3438%
	$x_2(55)$		C5C4DCCDD09B29BB FB2BD6FC3CF71698		3E12EDBA92939B9D B4F7C9CFD2DFBC84	
3	x_3	k_3	C96ED6BDA8250B23 E1AF940F6761F530	48.4375%	354DD0B808745D2E 3E3D0AF1B8513EBD	43.7500%
	$x_3(72)$		AFCDA22C41848DD0 EE8D5CD1386A7308		1DDF670C7F45FAB6 782B1C55AD698FAF	
4	x_4	k_4	869B983E6D567271 D6B4C0B39D625C47	53.9063%	13658FC810F73182 AB575F593C9399CB	50.7813%
	$x_4(101)$		24648020B2386BC0 8ADEB23A89153336		13AFD045C565EDDF 1454E73D95DD0A66	
5	x_5	k_5	40423E7FA52EA570 04B8E5369202070A	43.7500%	F405EF28A7FCB65B 0E9A1BA69352C192	51.5625%
	$x_5(127)$		BC0F1EBDAD3F9F3F 25E47C45228FD0CA		22B1B8AD09FECDDF D785A4123EE260B4	

4.2.4 Comparative analysis of cipher-text sensitivity

The process of cipher-text sensitivity analysis is as follows:

(i) Randomly produce a 256-bit key k and a plaintext x_1.
(ii) Use AES with k to encrypt x_1 to get the corresponding cipher-text y_1.
(iii) Randomly change y_1 by one-bit to get a new bit-sequence, denoted by y_2.
(iv) Use AES with k to decrypt y_2 to get the recovered bit-sequence, denoted by x_2.
(v) Calculate the index *diff* between x_1 and x_2.
(vi) Repeat the above steps 10,000 times to record the maximum, average and minimum values of *diff*.
(vii) Repeat the above processing with the unified algorithm instead of AES to obtain the maximum, average and minimum values of *diff* for the unified algorithm.

The test results according to the above method are listed in Table 8.

The last column in Table 8 shows the relative error between the average value of *diff* and the theoretical value 50%. As shown in Table 8, the calculated values of *diff* of the unified

Table 7 The calculation results of *diff* in plaintext sensitivity comparison analysis

Algorithm	*diff*			Relative error of the average value
	Maximum	Minimum	Average	
AES	68.7500%	35.1563%	50.0359%	0.0718%
Unified algorithm	67.9688%	34.3750%	49.9921%	0.0158%

Table 8 The calculation results of *diff* in cipher-text sensitivity comparison analysis

Algorithm	*diff*			Relative error of the average value
	Maximum	Minimum	Average	
AES	64.8438%	34.3750%	50.0427%	0.0854%
Unified algorithm	65.6250%	35.1563%	49.9860%	0.0280%

algorithm are similar to those of AES, and the relative errors are both less than 0.5%. Thus, the unified algorithm has strong cipher-text sensitivity as AES.

From the above analysis results in subsections 4.2.1–4.2.4, it can be seen that the unified algorithm has the same system sensitivity as AES. Thus, the unified algorithm possesses the same level of security intensity as AES. However, the unified algorithm has the advantage that AES does not have, that is, the unified algorithm has the same encryption process and decryption process, which can effectively save hardware resources, and simplify the secure communication protocol.

4.3 Cascade application

The AES algorithm usually works in cipher-text block chain (CBC) mode in symmetric encryption applications. Similarly, the unified algorithm can also work in CBC mode. For example, to encrypt a text of n bits, one can divide the text into m blocks of 128-bit length (padding 0 s if the last block's length is less than 128 bits), denoted by x_0, x_1,…,x_{m-1}, respectively, where m = Ceil(n/128), and Ceil(x) returns the smallest integer not less than x. Assume that the encryption key is denoted by k. The cascade system of unified algorithm is shown in Fig. 9. From Fig. 9, one can see that the cascade system is also unified, i.e. its encryption process and decryption process are the same.

In Fig. 9, "UA" represents the unified algorithm, "Flip" module flips the bit-sequence in bit (in microcontroller) or in byte (in computer), and "⊕" means bitwise exclusive-or operation. As can be seen from Fig. 9, the input $\{x_i\}$, i = 0,1,…,m-1, will be transformed into the output as follows:

$$y0 = \mathrm{UA}(x0), yi = \mathrm{UA}(xi \oplus \mathrm{Flip}(yi-1)) \oplus \mathrm{Flip}(xi-1), i = 1, 2, \ldots, m-1. \quad (4)$$

The system in Fig. 9 can be used to encrypt/decrypt a large amount of text data or image information. For example, it can be used to encrypt the image shown in Fig. 10a. The image

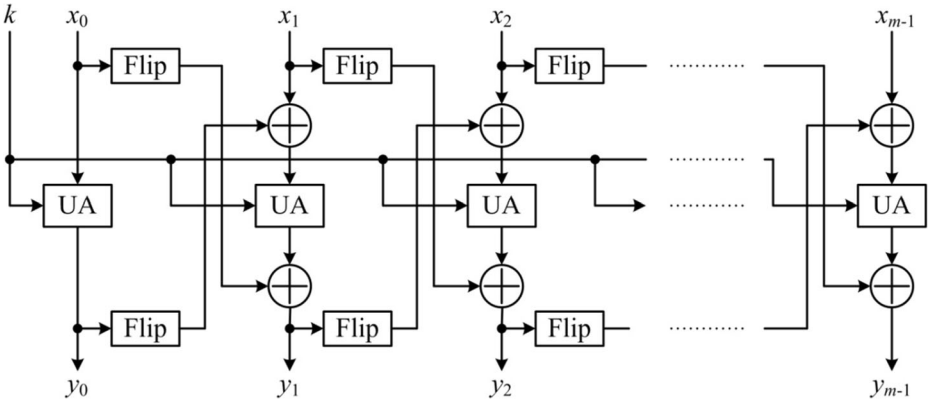


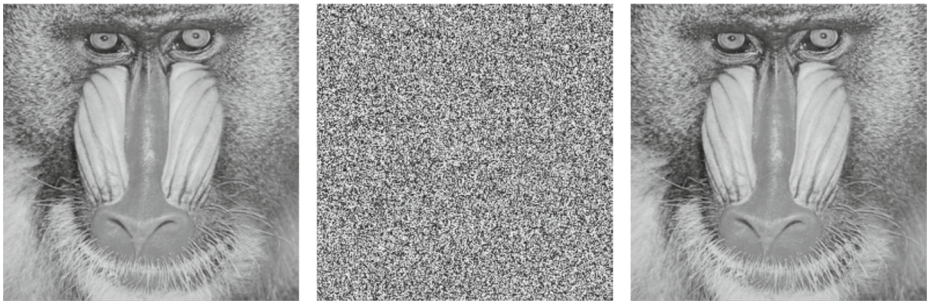
Fig. 9 The unified algorithm in CBC mode

Mandrill in Fig. 10a is an 8-bit grayscale image with the size of 256 × 256. At first, expand Mandrill row by row into a vector. Then, divide the vector into 4096 blocks of size 16 bytes (or 128 bits). That is, $m = 4096$ in Fig. 9. Then use the system in Fig. 9 with, for example, the k_1 in subsection 4.2.3, to encrypt the blocks to get the cipher blocks, named $\{y_i\}$, $i = 0,1,\ldots,4095$. Now, convert each y_i into 16 bytes and then reshape all the bytes into a 256 × 256-sized matrix, as shown in Fig. 10b. Figure 10b shows that the cipher image is a noise-like one with no visual patterns. Also, use the system in Fig. 9 to decrypt the image in Fig. 10b to get a recovered image, as shown in Fig. 10c, which is identical to the original image in Fig. 10a.

5 Conclusion

Based on the AES algorithm with a 256-bit key, this paper studies the unified algorithm with the same encryption and decryption processes. The unified algorithm uses the modules of "SubBytes", "MixColumns" and their inverse modules of AES, improves the module of "ShiftRows" and its inverse module of AES, adjusts the key expansion module of AES, and adds the flip module to the system. This paper employs the single-three program in Mathematica to compare and analyze the security performance of AES and the unified algorithm. The analysis results show that the unified algorithm possesses the processing speed of 1.6570Mbps, which is in the same level as the speed of AES. Like AES, the unified algorithm has strong sensitivity to the secret key, plaintext and cipher-text. Under the condition of comparable processing speed and security intensity with AES, the unified algorithm has the significant advantage of the same encryption process and decryption process, which can save the hardware resources of cryptographic system, and simplify the synchronous protocol in secure communications. The unified algorithm is an alternative for AES in various symmetric encryption applications.

This paper designs a unified algorithm based on AES with the 256-bit long key. In fact, the proposed unified algorithm is also applicable to AES with the 128-bit or 192-bit long key. When the key length is 128 bits and 192 bits, the AES algorithm will have 11 and 13 sub-keys, respectively. The corresponding unified algorithm will have 5 and 6 sub-keys, and the modified key expansion algorithm is still applicable. In the case of 128-bit long key, the unified algorithm will have five rounds with the shift values of "ShiftRows" being 3, 5, 7, 13 and 11 in turn. And in the case of 192-bit long key, the unified algorithm will have six rounds with the shift values of "ShiftRows" being 3, 5, 7, 9, 13 and 11 in turn. Thus, the unified algorithm can use 128-bit and 192-bit long keys.



(a) (b) (c)

Fig. 10 Result of image encryption based on the unified algorithm in CBC mode. **a** Mandrill; **b** The cipher image; **c** The decrypted image

Like AES, the unified algorithm can work in cipher-text block chain (CBC) mode to encrypt and decrypt long text or a large amount of data. This paper designs a system in CBC mode, which is composed of unified algorithm, exclusive-or operation, sequence flipping operation. In the system, the input and output of each level are fed to the next level, so that the system has the characteristics of unified algorithm, that is, the encryption process is exactly the same as the decryption process. Thus, the unified algorithm in CBC mode also saves half to the resources and simplifies the synchronous protocol in the network communications.

Acknowledgments This work was supported by the National Natural Science Foundation of China (No. 61762043), the Natural Science Foundation of Jiangxi Province, China (No. 20192BAB207022), and the Scientific Research Foundation of Jiangxi Provincial Education Department, China (No. GJJ190249, GJJ210507).

References

1. Artiles JAP, Chaves DPB, Pimentel C (2019) Image encryption using block cipher and chaotic sequences. Signal Process Image Commun 79:24–31
2. Basu S (2011) International data encryption algorithm (IDEA)–a typical illustration. J Glob Res Comput Sci 2(7):116–118
3. Bejo A, Adji TB (2018) The replacement of irreducible polynomial and affine mapping for the construction of a strong S-box. Nonlinear Dyn 93(4):2105–2118
4. Burr WE (2003) Selecting the advanced encryption standard. IEEE Secur Priv 1(2):43–52
5. Cheng S, Wang L, Ao N, Han Q (2020) A selective video encryption scheme based on coding characteristics. Symmetry 12(3):332
6. Chowdhary CL, Patel PV, Kathrotia KJ, Attique M, Perumal K, Ijaz MF (2020) Analytical study of hybrid techniques for image encryption and decryption. Sensors 20(18):5162
7. Daemen J, Rijmen V (2001) Reijndael: the advanced encryption standard. Dobb J Soft Tools Profess Program 26(3):137–139
8. Davis R (1978) The data encryption standard in perspective. IEEE Commun Soc Mag 16(6):5–9
9. Devi RR, Chamundeeswari VV (2020) Triple DES: privacy preserving in big data healthcare. Int J Parallel Prog 48(3):515–533
10. Dzwonkowski M, Rykaczewski R (2018) Secure quaternion Feistel cipher for DICOM images. IEEE Trans Image Process 28(1):371–380
11. Gupta SS, Chattopadhyay A, Sinha K, Maitra S, Sinha BP (2012) High-performance hardware implementation for RC4 stream cipher. IEEE Trans Comput 62(4):730–743
12. Haddad S, Coatrieux G, Moreau-Gaudry A, Cozic M (2020) Joint watermarking-encryption- JPEG-ls for medical image reliability control in encrypted and compressed domains. IEEE Trans Inf Forensics Secur 15:2556–2569
13. Komargodski I, Segev G (2020) From minicrypt to obfustopia via private-key functional encryption. J Cryptol 33:406–458
14. Kundi D, Khalid A, Aziz A, Wang C, Neill M, Liu W (2020) Resource-shared crypto-coprocessor of AES enc/dec with SHA-3. IEEE Trans Circuits Syst I 67(12):4869–4882
15. Langenberg B, Pham H, Steinwandt R (2020) Reducing the cost of implementing the advanced encryption standard as a quantum circuit. IEEE Trans Quantum Eng 1:1–12
16. Liao X, Li K, Yin J (2016) Separable data hiding in encrypted image based on compressive sensing and discrete Fourier transform. Multimed Tools Appl 75(11):13779–13789
17. Liao X, Yu Y, Li B, Li Z, Qin Z (2020) A new payload partition strategy in color image steganography. IEEE Trans Circ System Video Technol 30(3):685–696
18. Masoumi M (2019) A highly efficient and secure hardware implementation of the advanced encryption standard. J Inf Secur Appl 48:102371
19. Saravanan P, Kalpana P (2018) Novel reversible design of advanced encryption standard cryptographic algorithm for wireless sensor networks. Wirel Pers Commun 100(4):1427–1458
20. Seghier A, Li J, Sun D (2019) Advanced encryption standard based on key dependent S-Box cube. IET Inf Secur 13(6):552–558

21. Shah D, Shah T (2020) A novel discrete image encryption algorithm based on finite algebraic structures. Multimed Tools Appl 79(37):28023–28042
22. Shen Y, Guo C, Wang L (2020) Improved security bounds for generalized Feistel networks. IACR Trans Sym Cryptol 2020(1):425–457
23. Singh G, Kumar A, Sandha KS (2011) A study of new trends in Blowfish algorithm. Int J Eng Res Appl 1(2): 321–326
24. Stoyanov B, Nedzhibov G (2020) Symmetric key encryption based on rotation-translation equation. Symmetry 12(1):73
25. Sweatha AA, Pitchai KM (2020) Construction of cryptographically secure AES S-Box to second order reversible cellular automata. J Intel Fuzzy Syst 39(3):4313–4318
26. Wang C, Qiao S, Hei Y (2013) Low complexity implementation of block cipher SM4 algorithm. Comput Eng 39(7):177–180 (in Chinese)
27. Wang X, Li Y, Jin J (2020) A new one-dimensional chaotic system with applications in image encryption. Chaos, Solitons Fractals 139:110102
28. Yang CH, Chien YS (2020) FPGA implementation and design of a hybrid chaos-AES color image encryption algorithm. Symmetry 12(2):189
29. Yu W, Köse S (2017) A lightweight masked AES implementation for securing IoT against CPA attacks. IEEE Trans Circuits Syst I 64(11):2934–2944
30. Zhang Y (2017) A chaotic system based image encryption scheme with identical encryption and decryption algorithm. Chin J Electron 26(5):1022–1031
31. Zhang Y (2021) A new unified image encryption algorithm based on a lifting transformation and chaos. Inf Sci 547:307–327
32. Zhang X, Parhi KK (2002) Implementation approaches for the advanced encryption standard algorithm. IEEE Circ Syst Mag 2(4):24–46
33. Zhang Y, Tang Y (2018) A plaintext-related image encryption algorithm based on chaos. Multimed Tools Appl 77(6):6647–6669
34. Zhang Y, Chen A, Tang Y, Dang J, Wang G (2020) Plaintext-related image encryption algorithm based on perceptron-like network. Inf Sci 526:180–202
35. Zhou M, Wang C (2020) A novel image encryption scheme based on conservative hyperchaotic system and closed-loop diffusion between blocks. Signal Process 171:107484

Yong Zhang (Senior Member, IEEE) received the M.S. degree in communication and information systems and the Ph.D. degree in circuits and systems both form the University of Electronic Science and Technology of China, in 2003 and in 2006, respectively. He is a professor with the School of Software and Internet-of-Things Engineering and the director of Research Center for Quantum Computing in Jiangxi University of Finance and Economics in China. His research interests include cryptography and quantum computing. E-mail: zhangyong@jxufe.edu.cn

Aiguo Chen received the B.S. degree in computer science from Jiangxi Normal University in 1996, and the M.S. degree in management science and engineering from Jiangxi University of Finance and Economics in 2010. He is an assistant professor with School of Software and Internet-of-Things Engineering in Jiangxi University of Finance and Economics in China. His research interests focus on the cryptography and image processing. E-mail: chenaiguo@jxufe.edu.cn

Bin Chen received the M.S. degree and Ph.D. degree in signal and information processing both form the University of Electronic Science and Technology of China, in 2003 and in 2007, respectively. He is an associate professor with the School of Software and Internet-of-Things Engineering in Jiangxi University of Finance and Economics in China. His research interests focus on the chaos theory and cryptography. E-mail: chenbin@jxufe.edu.cn