22BCE3799

Apurba Koirala

Cryptography and Network Security Lab Assessment-4

AES operations

4

1. Key expansion

Code:

#include <iostream>

#include <vector>

#include <iomanip>

#include <sstream>

using namespace std;

const uint8\_t S[256] = {

0x63, 0x7C, 0x77, 0x7B, 0xF2, 0x6B, 0x6F, 0xC5, 0x30, 0x01, 0x67, 0x2B, 0xFE, 0xD7, 0xAB, 0x76,

0xCA, 0x82, 0xC9, 0x7D, 0xFA, 0x59, 0x47, 0xF0, 0xAD, 0xD4, 0xA2, 0xAF, 0x9C, 0xA4, 0x72, 0xC0,

0xB7, 0xFD, 0x93, 0x26, 0x36, 0x3F, 0xF7, 0xCC, 0x34, 0xA5, 0xE5, 0xF1, 0x71, 0xD8, 0x31, 0x15,

0x04, 0xC7, 0x23, 0xC3, 0x18, 0x96, 0x05, 0x9A, 0x07, 0x12, 0x80, 0xE2, 0xEB, 0x27, 0xB2, 0x75,

0x09, 0x83, 0x2C, 0x1A, 0x1B, 0x6E, 0x5A, 0xA0, 0x52, 0x3B, 0xD6, 0xB3, 0x29, 0xE3, 0x2F, 0x84,

0x53, 0xD1, 0x00, 0xED, 0x20, 0xFC, 0xB1, 0x5B, 0x6A, 0xCB, 0xBE, 0x39, 0x4A, 0x4C, 0x58, 0xCF,

0xD0, 0xEF, 0xAA, 0xFB, 0x43, 0x4D, 0x33, 0x85, 0x45, 0xF9, 0x02, 0x7F, 0x50, 0x3C, 0x9F, 0xA8,

0x51, 0xA3, 0x40, 0x8F, 0x92, 0x9D, 0x38, 0xF5, 0xBC, 0xB6, 0xDA, 0x21, 0x10, 0xFF, 0xF3, 0xD2,

0xCD, 0x0C, 0x13, 0xEC, 0x5F, 0x97, 0x44, 0x17, 0xC4, 0xA7, 0x7E, 0x3D, 0x64, 0x5D, 0x19, 0x73,

0x60, 0x81, 0x4F, 0xDC, 0x22, 0x2A, 0x90, 0x88, 0x46, 0xEE, 0xB8, 0x14, 0xDE, 0x5E, 0x0B, 0xDB,

0xE0, 0x32, 0x3A, 0x0A, 0x49, 0x06, 0x24, 0x5C, 0xC2, 0xD3, 0xAC, 0x62, 0x91, 0x95, 0xE4, 0x79,

0xE7, 0xC8, 0x37, 0x6D, 0x8D, 0xD5, 0x4E, 0xA9, 0x6C, 0x56, 0xF4, 0xEA, 0x65, 0x7A, 0xAE, 0x08,

0xBA, 0x78, 0x25, 0x2E, 0x1C, 0xA6, 0xB4, 0xC6, 0xE8, 0xDD, 0x74, 0x1F, 0x4B, 0xBD, 0x8B, 0x8A,

0x70, 0x3E, 0xB5, 0x66, 0x48, 0x03, 0xF6, 0x0E, 0x61, 0x35, 0x57, 0xB9, 0x86, 0xC1, 0x1D, 0x9E,

0xE1, 0xF8, 0x98, 0x11, 0x69, 0xD9, 0x8E, 0x94, 0x9B, 0x1E, 0x87, 0xE9, 0xCE, 0x55, 0x28, 0xDF,

0x8C, 0xA1, 0x89, 0x0D, 0xBF, 0xE6, 0x42, 0x68, 0x41, 0x99, 0x2D, 0x0F, 0xB0, 0x54, 0xBB, 0x16

};

const uint32\_t RC[10] = {

0x01000000, 0x02000000, 0x04000000, 0x08000000,

0x10000000, 0x20000000, 0x40000000, 0x80000000,

0x1B000000, 0x36000000

};

uint32\_t g(uint32\_t w) {

return (w << 8) | (w >> 24);

}

uint32\_t SubWord(uint32\_t w) {

return (S[(w >> 24) & 0xFF] << 24) |

(S[(w >> 16) & 0xFF] << 16) |

(S[(w >> 8) & 0xFF] << 8) |

(S[w & 0xFF]);

}

void ExpandKey(const uint8\_t\* k, uint32\_t\* w, int Nk, int Nr) {

int i = 0;

while (i < Nk) {

w[i] = (k[4 \* i] << 24) | (k[4 \* i + 1] << 16) |

(k[4 \* i + 2] << 8) | (k[4 \* i + 3]);

i++;

}

i = Nk;

while (i < 4 \* (Nr + 1)) {

uint32\_t temp = w[i - 1];

if (i % Nk == 0) {

temp = SubWord(g(temp)) ^ RC[i / Nk - 1];

}

w[i] = w[i - Nk] ^ temp;

cout << "w[" << i << "]: " << hex << setw(8) << setfill('0') << w[i] << endl;

i++;

}

}

void PrintExpandedKey(uint32\_t\* w, int Nr) {

for (int i = 0; i < 4 \* (Nr + 1); i++) {

cout << hex << setw(8) << setfill('0') << w[i] << " ";

if ((i + 1) % 4 == 0)

cout << endl;

}

}

int main() {

uint8\_t k[16];

string keyInput;

cout << "Enter 16-byte AES key in hexadecimal";

cin >> keyInput;

if (keyInput.length() != 32) {

cout << "Invalid input length! Expected 32 hex characters (16 bytes)." << endl;

return 1;

}

for (int i = 0; i < 16; i++) {

stringstream ss;

ss << hex << keyInput.substr(i \* 2, 2);

int byte;

ss >> byte;

k[i] = static\_cast<uint8\_t>(byte);

}

int Nk = 4;

int Nr = 10;

uint32\_t w[44];

ExpandKey(k, w, Nk, Nr);

cout << "\nExpanded Key Schedule:\n";

PrintExpandedKey(w, Nr);

return 0;

}

Output

A screen shot of a computer code

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

1. Shift rows

#include <iostream>

using namespace std;

const int N = 4;

void shiftRows(unsigned char state[N][N]) {

unsigned char temp;

temp = state[1][0];

for (int i = 0; i < N - 1; i++)

state[1][i] = state[1][i + 1];

state[1][N - 1] = temp;

temp = state[2][0];

state[2][0] = state[2][2];

state[2][2] = temp;

temp = state[2][1];

state[2][1] = state[2][3];

state[2][3] = temp;

temp = state[3][N - 1];

for (int i = N - 1; i > 0; i--)

state[3][i] = state[3][i - 1];

state[3][0] = temp;

}

void printState(unsigned char state[N][N]) {

for (int i = 0; i < N; i++) {

for (int j = 0; j < N; j++) {

printf("%02X ", state[i][j]);

}

cout << endl;

}

}

int main() {

unsigned char state[N][N];

cout << "Enter 16 bytes (in hex) for the AES state matrix (row-wise):\n";

for (int i = 0; i < N; i++) {

for (int j = 0; j < N; j++) {

int val;

cin >> hex >> val;

state[i][j] = static\_cast<unsigned char>(val);

}

}

cout << "\nOriginal State Matrix:\n";

printState(state);

shiftRows(state);

cout << "\nState Matrix After ShiftRows:\n";

printState(state);

return 0;

}

Output:

A screenshot of a computer

AI-generated content may be incorrect.

1. Mix columns

#include <iostream>

using namespace std;

const int N = 4;

unsigned char gmul(unsigned char a, unsigned char b) {

unsigned char p = 0;

for (int i = 0; i < 8; i++) {

if (b & 1)

p ^= a;

bool hiBitSet = (a & 0x80);

a <<= 1;

if (hiBitSet)

a ^= 0x1B;

b >>= 1;

}

return p;

}

void mixColumns(unsigned char state[N][N]) {

unsigned char temp[N][N];

for (int c = 0; c < N; c++) {

temp[0][c] = gmul(0x02, state[0][c]) ^ gmul(0x03, state[1][c]) ^ state[2][c] ^ state[3][c];

temp[1][c] = state[0][c] ^ gmul(0x02, state[1][c]) ^ gmul(0x03, state[2][c]) ^ state[3][c];

temp[2][c] = state[0][c] ^ state[1][c] ^ gmul(0x02, state[2][c]) ^ gmul(0x03, state[3][c]);

temp[3][c] = gmul(0x03, state[0][c]) ^ state[1][c] ^ state[2][c] ^ gmul(0x02, state[3][c]);

}

for (int i = 0; i < N; i++)

for (int j = 0; j < N; j++)

state[i][j] = temp[i][j];

}

void printState(unsigned char state[N][N]) {

for (int i = 0; i < N; i++) {

for (int j = 0; j < N; j++) {

printf("%02X ", state[i][j]);

}

cout << endl;

}

}

int main() {

unsigned char state[N][N];

cout << "Enter 16 bytes (in hex) for the AES state matrix (row-wise):\n";

for (int i = 0; i < N; i++) {

for (int j = 0; j < N; j++) {

int val;

cin >> hex >> val;

state[i][j] = static\_cast<unsigned char>(val);

}

}

cout << "\nOriginal State Matrix:\n";

printState(state);

mixColumns(state);

cout << "\nState Matrix After MixColumns:\n";

printState(state);

return 0;

}

Output:

A screenshot of a computer

AI-generated content may be incorrect.