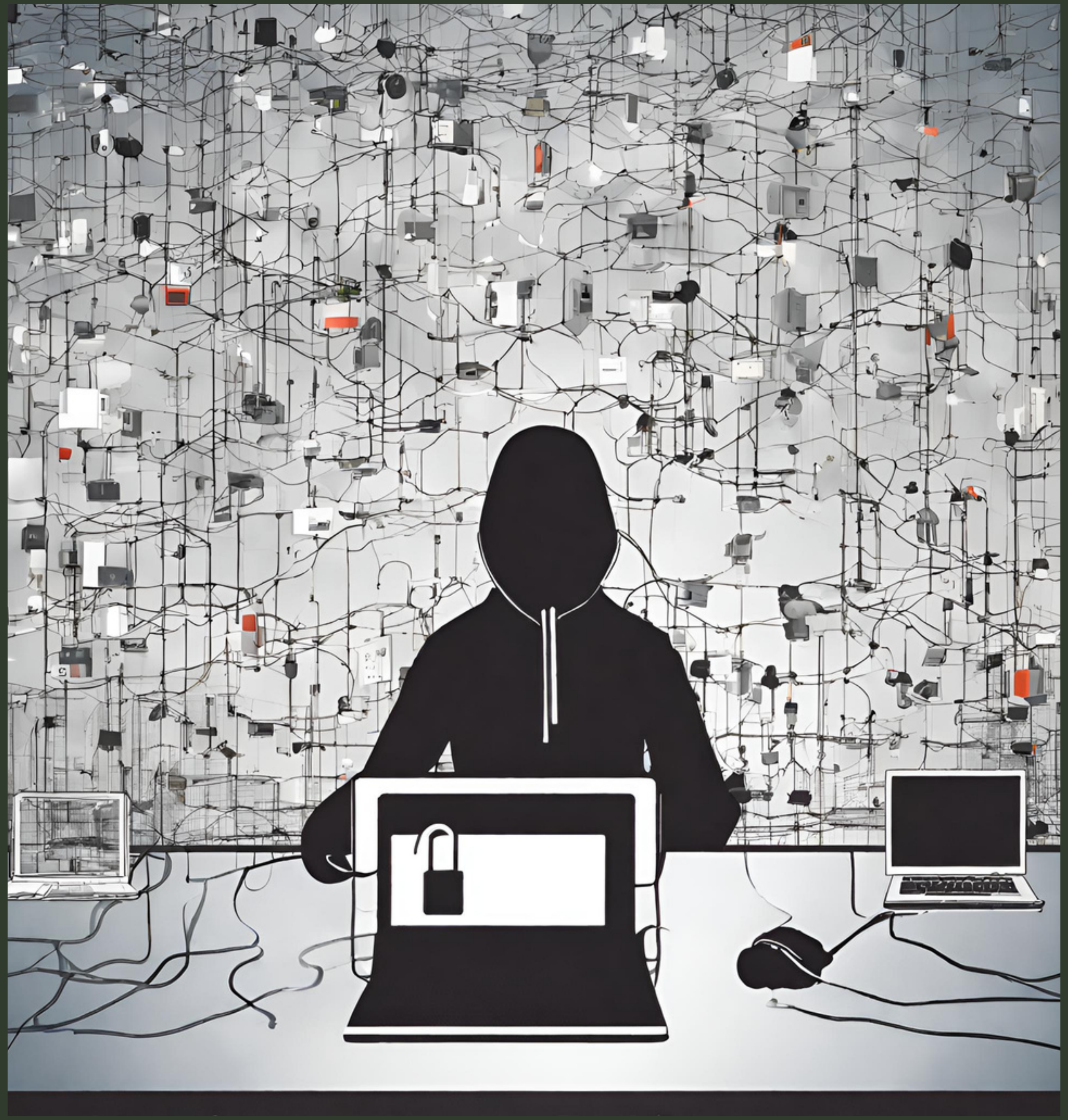
Feb 22, 2024

Assignment 2 Project - 1

Presented by Sarvagya Kaushik & Kunal Sharma

Winter 2024



Network Security

Prof. B N Jain

int main(): The main() function reads plaintext data from a file named ‘input.txt’, converts it into hexadecimal format, encrypts it using AES, and then decrypts the ciphertext back to plaintext. It iterates this process three times for different input data. The encryption and decryption are performed using AES encryption and decryption functions respectively, with a predefined encryption key. Finally, it prints out the ciphertext and decrypted plaintext for each iteration.

vector<vector<uint8_t>> encrypt(vector<vector<string>> plaintext_matrix_hex, vector<vector <uint8_t>>& ciphertext_matrix_hex,int r_number, vector<vector<uint8_t>> &key) : This function performs AES encryption on a plaintext matrix. It begins with an initial Add Round operation, where each element of the plaintext matrix is XORed with the corresponding element of the encryption key to produce the ciphertext matrix. Then, it iterates through 10 rounds of encryption using the encrypt_round() function, updating the ciphertext matrix accordingly and printing the output of the 1st and 9th encryption rounds.

vector<vector<uint8_t>>encrypt_round(vector<vector<uint8_t>> plaintext_matrix_he, vector< vector<uint8_t>>ciphertext_matrix_hex,int r_number, vector<vector<uint8_t>> &prev_r_key): This function represents one round of AES encryption. It performs substitution and row-shifting operations on the plaintext matrix to produce the ciphertext matrix. If it's not the final round, it additionally applies the MixColumns operation by multiplying a fixed matrix with the current state of ciphertext. At last, it combines the result with the round key using XOR. If it's the final round, it only combines the ciphertext with the round key.

uint8_t substitution(**uint8_t num**) : This function performs a substitution operation on an 8-bit integer 'num' using a predefined matrix multiplication and addition operation. It first calculates the multiplicative inverse of num using the m_inv() function. Then, it initializes two matrices matrixA and matrixB and converts the multiplicative inverse into a bitset. The function then performs matrix multiplication and addition using matrixMultiplicationAndAddition() function. At last, it converts the resulting bitset back to an integer and returns.

vector<int> matrixMultiplicationAndAddition(**const int matrixA**[][**8**], **const int matrixB**[], **const vector<int> &matrixC**) **:** This function performs matrix multiplication between ‘matrixA’ and ‘matrixC’, followed by element-wise addition with ‘matrixB’. It returns a vector named ‘result’ containing the resulting values. The multiplication is achieved by bitwise AND between corresponding elements of rows of ‘matrixA’ and ‘matrixC’, followed by XOR operations to accumulate the results. Then, each element of ‘matrixB’ is XORed with the corresponding element of the result vector to perform addition.

void rotate(**vector<uint8_t> &vec, int d**) **:** This function rotates the elements of a vector ‘vec’ to the left by ‘d’ positions. It achieves this by repeatedly moving the first element to the end of the vector ‘d’ times.

uint8_t gf_multiply(**uint8_t a, uint8_t b**) **:** This function does Galois Field multiplication between two 8-bit integers ‘a’ and ‘b’. It iterates through the bits of ‘b’, XORs ‘a’ with the result if the least significant bit of ‘b’ is set, and updates ‘a’ according to the Galois Field polynomial. The final result is returned.

void create_round_key(vector<vector<uint8_t>> &prev_round_key, int round_number): This function creates the round key for AES encryption based on the previous round key and the current round number. It generates a temporary vector ‘vec’ by extracting the last column of the previous round key, applies the ‘g’ function using g_func(), and then updates the round key by XORing each element with either the corresponding element of ‘vec’ or the element of the previous column.

void g_func(vector<uint8_t> &vec, int round_number) : This function performs the ‘g’ function used in AES encryption for a given round. It rotates the input vector, substitutes its elements, and finally XORs the first element with the round constant calculated using RC function.

vector<vector<uint8_t>> decrypt(vector<vector<string>> ciphertext_matrix_hex, vector<vector <uint8_t>> &plaintext_matrix_hex, int r_number, vector<vector<uint8_t>> &key) : This function performs AES decryption on a ciphertext matrix. It first converts the ciphertext matrix to byte format. Then, it iterates through 10 rounds of decryption using the decrypt_round function, updating the plaintext matrix accordingly and optionally printing the output of the 1st and 9th decryption rounds. Finally, it applies the last Add Round operation by combining the result with the final round key.

vector<vector<uint8_t>> decrypt_round(vector<vector<uint8_t>> plaintext_matrix_he, vector <vector<uint8_t>> ciphertext_matrix_hex, int r_number, vector<vector<uint8_t>>&prev_r_key): This function represents one round of AES decryption. It starts by combining the ciphertext matrix with the round key using XOR. If it's not the first round, it then performs the Inverse MixColumns operation by multiplying the fixed inverse matrix with the current state of ciphertext. Next, it applies the Inverse ShiftRows operation by rotating rows in the opposite direction. Finally, it applies the Inverse Substitution operation to the ciphertext matrix to obtain the plaintext matrix for that round.

void inv_create_round_key(vector<vector<uint8_t>>&prev_round_key,int r_number): This function creates the round key for AES decryption based on the previous round key and the current round number. It iteratively XORs elements of the previous round key columns either with the corresponding element from a temporary vector ‘vec’ or the element from the previous column. Before updating each column, it constructs ‘vec’ from the last column of the previous round key and applies the ‘g’ function in reverse order.

uint8_t RC(int x) : This function calculates the round constant for AES encryption using recursion. If the input 'x' is 0x01, it returns 0x01; otherwise, it recursively calculates the round constant by multiplying '0x02' with the round constant of the previous round (x - 1).

uint8_t inv_substitution(uint8_t num) : This function performs an inverse substitution operation on an 8-bit integer ‘num’. It initializes two matrices ‘matrixA’ and ‘matrixB’, converts ‘num’ into a bitset, and performs matrix multiplication and addition using the matrixMultiplicationAndAddition function(). Then, it converts the resulting bitset back to an integer, calculates the multiplicative inverse using m_inv() function, and returns the result.

uint8_t m_inv(uint8_t x) : This function calculates the multiplicative inverse of a given 8-bit integer ‘x’ within a Galois Field. It iterates through possible values of ‘i’, computing the product of ‘x’ and ‘i’ using the gf_multiply() function and returns the first ‘i’ for which the product is equal to 0x01, indicating the multiplicative inverse.

Thank you !

Name	Sarvagya Kaushik
Roll No.	2021350
Branch	CSD
E-mail	Sarvagya21350@iiitd.ac.in

Name	Kunal Sharma
Roll No.	2021331
Branch	CSD
E-mail	Kunal21331@iiitd.ac.in

Winter 2024