DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Indian Institute of Technology Bhilai

CS553/CSL505 – CRYPTOGRAPHY

Semester: 2024-M

Scope: AES, Integral Cryptanalysis, Mode of Operation, Hash Functions

Assignment 4

November 12, 2024

- Instructions
  - LaTeX based answers are preferred *(Use a single LaTeX file to answer all questions)*
  - "Readme" file for your code (if applicable)
  - Submissions in a zip file named as `<group-name>_<assignment_no>`

1. Encode your name (including spaces) in the AES state matrix as shown in class.
   - If your name has greater than 14 characters (including spaces) then use only the first 12 characters.
   - Pad the rest of the state to make it 16 bytes.
   - For padding use (and mention in the answer) any scheme from `https://en.wikipedia.org/wiki/Padding_(cryptography)`
   - Now apply the `ShiftRows` operation on the state.
   - Next apply the `SubBytes` operation.
   - Show the state matrix after every operation.
2. - Write a code to generate the DDT of `AES`-SBox in your favorite programming language.
   - Retrieve the following info from the DDT using a function for each:
     - Number of zeros in the table
     - Number of 4's per fixed input difference
     - Number of 4's per fixed output difference
3. - Using your favorite programming language implement the `AES` key-expansion algorithm
   - Now use the state matrix initialized with your name in Problem 1 as your initial key
   - Show the 10 rounds keys in the main assignment[1].
   - For the fifth round-key show all the steps of the key-expansion algorithm that leads to the sixth round key.
4. - Consider that there are two rounds of `AES`
     - Using `openssl` generate a 128-bit message to initialize the `AES` state matrix.
     - Now modify any byte of this state to get another state.
     - Run two rounds of AES on these states

[1] You can dump the latex tables from within your code to make life easier.

    - For rounds keys you can use output of Problem-3
    - After every operation take the XOR of the states
    - In the main assignment show the two states used. Then show how the XOR of the states propagates though each step of the round.
    - Verify if you can observe a distinguishing pattern in the output.
    - Write a code for the above[2]
- Also see how the pattern changes as you change the location of the modified byte for the second state.

5.
- In your favorite programming language verify the 3-round integral distinguisher on `AES`.
- You code should verify the properties after every round displaying necessary messages.
- You should have sub-routines like `isAll`, `isConstant`, `isBalanced`.
- Also the constant part of the 256 states should be randomly generated to demonstrate that it has no effect on the result.
- You don't need show anything in main assignment for this, just a few lines on how to test you code.

6.
- Use your implementation of Sypher004 in that you did in one of the earlier assignments.
- Implement its decryption.
- Now implement CBC and CFB modes of operation shown in class with the encryption and decryption of Sypher004.
- For CFB use $t = 4$.
- Assume that your test message is a multiple of 16-bits.[3]
- Now simulate error in transmission by flipping some bits in one of the cipher-text blocks.
- Show the error propagation in the decrypted message by comparing it with original message.

7.
- Find out why the IV for CTR can be a nonce
- While for CBC it must be randomized (unpredictable)
- Give short justification for both

8.
- Book: Serious Cryptography
- Implement Listing 7-1/7-2 and share the results you get after running 7-2.
- Now replace the compare function in 7-1 with the one defined in Listing 7-3 and rerun 7-1. Share the results.
- You stats should be accompanied by the screen-shots of the actual run.

9.
- Visit `https://malicioussha1.github.io/`
- Find out what is the vulnerability.

[2] Be careful while implementing MixColumns

[3] So no padding required