RETRACING BOOMERANG ATTACK

The Retracing Boomerang Attack is a key-recovery attack that builds on the classical boomerang attack framework by using differential properties of a reduced-round cipher and exploiting round symmetry to derive linear constraints on key material. It is particularly well-suited for reduced-round versions of AES.

Steps:

1. Use a distinguisher to find a good pair of plaintexts.
2. Generate the friend pairs.
3. For each friend pair, generate a linear equation using the AES MixColumns.
4. Construct the matrix A from all linear equations.
5. Perform gaussian elimination to solve the system and recover the key

Files

aes.cpp: Runs AES encryption and decryption algorithm with flexible number of rounds.

yoyo.cpp: Generate the good pairs of plaintexts. A **good pair** is a plaintext pair whose differences propagate through a reduced-round cipher in a controlled way.

retracing.cpp: Generate the good pairs using yoyo.cpp and generate 1034 friend pairs. Construct the linear equation using the friend pairs and MixColumns and finds the rank of the matrix.

The folder include contains all the header files required to run source files.

The folder tests contains the test program to test the source files with the inputs

test_aes.cpp: checks whether aes.cpp works or not. This is done by giving plaintext and expected ciphertext pairs with key 0, key and expected ciphertext pairs with plaintext 0, test the encryption function with one bit change in the key, test the encryption function with one bit change in the plaintext.

test_retracing_boomerang.cpp: Runs the retracing boomerang attack on 5 round AES using retracing.cpp. Test is repeated for 100 random keys.

test_yoyo_pass.cpp/test_yoyo_fail.cpp: checks yoyo distinguisher (persent in yoyo.cpp) for 10 times and check whether it returns good pair of plaintexts or not.

Execution

1. $mkdir build
2. $cmake
3. $cmake –build
4. $ctest

ctest runs all the test files and check whether all the tests are passed or not

test_aes.cpp
Input: (Plaintext, ciphertext) pairs, (key, ciphertext) pairs, Plaintext, ciphertext) pairs with one bit change in the plaintext, (key, ciphertext) pairs with one bit change in the key

output: passed/failed

test_retracing_boomerang.cpp:

Input: 100 random keys (created using random function)

Output: passed/failed

test_yoyo_pass.cpp:

Input: 10 random keys (created using random function)

Output: passed/failed (passed: good pair is found otherwise failed)

test_yoyo_fail.cpp:

Input: 10 random keys (created using random function)

Output: passed/failed (passed: no good pair is found, failed otherwise)

Future: This code generates the system of linear equation. Solving the linear equation, finding the key from the solution is yet to be completed.