Kyan Sharma November 4, 2024

CS-1306 / MAT-1201 Problem Set 1

Collaborators: *none*

Problem 1

i) Having to custody and compute the keyshares wasn't a great process at all. Given the fact that it was fairly bruteforced in such a short time frame gives away that it wasn't a good measure at all. Lets look at all the Flaws bit by bit(literally)
1. Happy's Claim 1: "12800 bits superiority"
If we look closely the effective key size if we have all the 100 random xor would just end up being
C(100,2) which is effectively 4950 possible key pairs for us to bruteforce
log2(4950) which is roughly 12.27 bits over here
Which is way smaller and no where close to the initial 128 bits that the traditional AES system offers.
Also the key over here is derived from simple XOR operations but since we only need two key shares to compute
the final key which ends up leading to a massive decrease in entropy and highly susceptible to brute force attacks. 2. Happy's Claim 2: "Safe to store key shares on disk"
This shows Happys Claim of Security is flawed firstly purely relying on the facade of randomness the keys give is not security the relationship between these is what makes a system secure not their individual randomness. The entire key material should be protected not only the final keys. Since we saw that if attacker gains access we can just bruteforce all 4950 operations fairly quickly and no additional security measures implemented. 3: Happy's Claim "Easy to remember indices like ATM PIN" Looking at both the security and usability problems over here
Limited range over here (0-99) makes guessing fairly easy and people choose memorable patterns usually in such cases, will be weak against social engineering attacks apart from brute force attempts.
From a usability pov there is no recovery mechanism over here if indices are forgotten apart from bruteforce.

Snake-oil Implementation Analysis: 1. Padding Vulnerabilities: Zero padding scheme leaves it open to padding oracle attacks where if size of blocks are known we can extract the information over here where they send modified ciphertext to the server and they learn based on responses if the padding is valid or invalid and then recover the plaintext back from it because it is in CBC mode too Since it lacks the integrity checks over here.

Also using CBC mode without authentication is a serious concern because it lacks data integrity and authenticity mechanism to see if the ciphertext has been altered or not and suspecitible to bit flipping and replay attacks. (In banking system can end up making the bank send payment to their account without detection)
Also over here it lacks methods for key rotation, does not generate the keys securely and has a weak key share distribution method.

Programming Implementation Issues of snakeoil: The error messages in "AES128::decrypt" function could leak information about internal state potentially helping in cryptanalysis attempts. It also has no logging mechanisms providing so there is no way to track if any attacker is trying to attack the system or audit such attacks.
Also this implementation over here does not securely wipe memory after key operations where memory dumps could end up recovering keys, lack of memory access pattern obfuscation makes it susceptible to side channel attacks too. Which provides many attack vectors without even the bruteforce part.
Vulnerable to frequency analysis and no protection against plain text attacks.

Positive aspects: Its Simple implementation which is quick to deploy,relatively easy to understand and doesn't have a lot of external dependencies.

Negative aspects:
Mentioned most of it before no backup mechanism(password recovery) and has poor scalability too.

Overall:
While attempting to improve upon AES-128, it ends up numerous vulnerabilities and reverse ends up providing significantly less security with a lot of flaws in the cryptosystem. It reflects the poor understanding of basic cryptographic principles and is a warning to implementing such custom protocols without rigorous sound texting and should stick to standard well vetted cryptography solutions.

Problem 2

I replaced it with the OpenSSl Crypto because the Botan version wasnt working for me. And fitted it with the zero padding instead of the standard PKCS7 which made me face issues initially
I coded the functions and modified the AES.cpp,AES.hpp,analyze.cpp,crypto.cpp,crypto.hpp,snakeoil.cpp,M

file because of the change in the botan implementation.
Yes it decrypts if you run make clean and make all to key 4 and 31 over here the sample.enc on merrywives.dat file.

Problem 3

For space.dat I ran all the mystery.enc on space.dat and got the following mystery1.enc:
Guessed key:
Indices 14 and 93
a9 f3 60 b8 ff 39 60 8c
a6 fb 9f 44 42 7c 6a 8a
Four score and seven years ago our fathers brought forth on this continent, a new nation, conceived in Liberty, and dedicated to the proposition that all men are created equal.

Now we are engaged in a great civil war, testing whether that nation, or any nation so conceived and so dedicated, can long endure. We are met on a great battle-field of that war. We have come to dedicate a portion of that field, as a final resting place for those who here gave their lives that that nation might live. It is altogether fitting and proper that we should do this.

But, in a larger sense, we can not dedicate – we can not consecrate – we can not hallow – this ground. The brave men, living and dead, who struggled here, have consecrated it, far above our poor power to add or detract. The world will little note, nor long remember what we say here, but it can never forget what they did here. It is for us the living, rather, to be dedicated here to the unfinished work which they who fought here have thus far so nobly advanced. It is rather for us to be here dedicated to the great task remaining before us – that from these honored dead we take increased devotion to that cause for which they gave the last full measure of devotion – that we here highly resolve that these dead shall not have died in vain – that this nation, under God, shall have a new birth of freedom – and that government of the people, by the people, for the people, shall not perish from the earth.
mystery2.enc
Guessed key:
Indices 55 and 82
33 43 ff cd c0 2e 07 cc
aa 8a ec 88 59 2e e0 34
I have a dream that one day this nation will rise up and live out the true meaning of its creed: "We hold these truths to be self-evident: that all men are created equal." I have a dream that one day on the red hills of Georgia the sons of former slaves and the sons of former slaveowners will be able to sit down together at a table of brotherhood. I have a dream that one day even the state of Mississippi, a desert state, sweltering with the heat of

injustice and oppression, will be transformed into an oasis of freedom and justice. I have a dream that my four children will one day live in a nation where they will not be judged by the color of their skin but by the content of their character. I have a dream today.
mystery3.enc
Guessed Key
Indices 2 and 56
45 68 6d 71 c5 ac 75 8f
6b 4b d2 a8 f7 18 f6 c8
That's one small step for man; one giant leap for mankind. mystery4.enc
Guessed key:
Indices 1 and 97
d9 5d 75 43 b0 f6 d3 0d
99 fa 5a 27 10 ea 0f 26
lk .— (garbage text unsuccesful)
sample.enc
Guessed key: Indices 4 and 31
e7 8e bb 99 44 41 f6 ec
33 d7 5c fe bb e7 00 49
This is a sample text that is encrypted using keyshares with key indices 4 and 31.

For uniform.dat I ran all the mystery.enc on uniform.dat and got the following

mystery1.enc
Guessed key: Indices 43 and 65 30 0c 3e d9 c6 11 06 82 55 7a 2a b3 f6 f0 9f 36

garbage that doesn't compile in latex its on the github though

mystery2.enc
Guessed key: Indices 6 and 35 bd 02 fd 58 61 99 17 e0 4c 66 c4 f0 ef 9e 89 e7

garbage that doesn't compile in latex its on the github though mystery3.enc
Guessed key:
Indices 3 and 18
b8 7b 9c 10 61 f0 75 3d
52 b8 30 dc 76 58 5a 08
garbage that doesn't compile in latex its on the github though
mystery4.enc
Guessed key:
Indices 0 and 1
58 49 e0 b9 24 ca 74 aa
27 0a ee 30 79 d2 62 18
garbage that doesn't compile in latex its on the github though
sample.enc

Guessed key:
Indices 7 and 43
f7 16 a0 7a 84 d7 57 f8
61 4d 8e ae c0 13 0d 8f
garbage that doesn't compile in latex its on the github though

For merrywives.dat I ran all the mystery.enc on uniform.dat and got the following

mystery1.enc
Guessed key:
Indices 14 and 93
a9 f3 60 b8 ff 39 60 8c
a6 fb 9f 44 42 7c 6a 8a
Same decryption as space.dat of mystery1.enc
mystery2.enc
Guessed key: Indices 55 and 82
33 43 ff cd c0 2e 07 cc
aa 8a ec 88 59 2e e0 34
Same decryption as space.dat of mystery2.enc

mystery3.enc
Guessed key:
Indices 2 and 56
45 68 6d 71 c5 ac 75 8f
6b 4b d2 a8 f7 18 f6 c8
Same decryption as space.dat of mystery3.enc

mystery4.enc
Guessed key: Indices 13 and 82 c2 40 63 20 25 0e 73 8b af 54 a7 d8 f1 fb 66 4a garbage that doesn't compile in latex its on the github though

sample.enc
Guessed key:
Indices 4 and 31
e7 8e bb 99 44 41 f6 ec
33 d7 5c fe bb e7 00 49
Same as space.dat of sample.enc

For ulysses.dat I ran all the mystery.enc on uniform.dat and got the following

mystery1.enc
Guessed key:
Indices 14 and 93
a9 f3 60 b8 ff 39 60 8c
a6 fb 9f 44 42 7c 6a 8a
Same decryption as space.dat of mystery1.enc

mystery2.enc
Guessed key: Indices 55 and 82
33 43 ff cd c0 2e 07 cc
aa 8a ec 88 59 2e e0 34
Same decryption as space.dat of mystery2.enc

mystery3.enc
Guessed key:
Indices 2 and 56
45 68 6d 71 c5 ac 75 8f
6b 4b d2 a8 f7 18 f6 c8
Same decryption as space.dat of mystery3.enc

mystery4.enc
Guessed key:
Indices 13 and 82
c2 40 63 20 25 0e 73 8b
af 54 a7 d8 f1 fb 66 4a garbage that doesn't compile in latex its on the github though

sample.enc
Guessed key:
Indices 4 and 31
e7 8e bb 99 44 41 f6 ec
33 d7 5c fe bb e7 00 49
Same as space.dat of sample.enc

So this was all the decryptions and anything that is in english understanding was succesful and the garbage ciphertext ones were not all the files of these are available on my github

repo too.
https://github.com/Kyan148369/AES_Encryption_Decryption

We succesully decryopted 4/5 files from all except uniform.dat which gave 0/5 files .

Difference in lengths did not have any impact in the success of the decryption here.
english based frequency tables all performed similiar here where uniform distribution failed as expected.
We can also conclude that mystery4.enc is some form of uniform character distribution that is unaffectted by these 4 distributions.

So for constructing the uncrackable message:
We will use special characters that dont appear much in english, double the message to prevent key duration from playing in our favour and have similiar divergence scores for all of these I just put some random special characters and it failed to decrypt for all 3 data tables over here saved as mystery5.enc in my github