Homework 6 Hard Copy

Christopher Chan
Chan328

Code Explanation Part 1:

For the RSA implementation, I first started by creating a function that would generate prime values for ‘p’ and ‘q’ while referencing the gcd of p and ‘e’ and the q and ‘e’. Those values are then written to text files for storage. After generating p and q, the encrypt function would take those p and q values from those text files and get a value for ‘n’ to use in the encryption process. Blocks of 128 bits are read from the plaintext file and padded with 0s on the right if the read block wasn’t completely 128 bits long. This block is then padded with 0s on the right to make it 256 bits for the encryption process. The 256 bit block is then encrypted by doing the block to the power of e mod n. This process is repeated for encryption until all the plaintext is read and the combined encrypted text is written to the output file. For decryption, this is where things are a bit more complicated. The starter steps are pretty similar to encryption, reading p and q from their text files and generating n with those values. Things then change when the totient is calculated by following the equation (p-1)(q-1) and this totient is used to find the d using e and finding its multiplicative inverse. Blocks of 256 bits are then read from the ciphertext and equations from lecture 12.5 are referenced, finding Vp, Vq, Xp and Xq. After this, if we apply the Chinese Remainder Theorem, we can get the decrypted block and strip the padded zeros from it. The combined plaintext is written to the decrypted file.

Code Explanation Part 2:

The break RSA function program uses a lot of the RSA implementation in terms of functionality but just does it for 3 different distinct values of p and q. The p and q generation as well as the encrypt were modified so that p and q didn’t need to be written to text files and the p and q generation was instead called at the beginning of the encrypt function. Three different n values were calculated and the encrypt function from the RSA was called to do the encryption for those 3 values. The 3 separate encrypted texts were then written to their own encrypted files, as well as the 3 different n’s used to their own text file. The crack function is the portion that was quite different to my RSA implementation. I first read in and separated out the 3 different n values I used to encrypt from the text file it was in. Then using the equations from lecture 11.7 on the Chinese remainder theorem. I followed the listed equations to find values for ci using the different Ms I was able to calculate from the n values created in the encryption process. I then read in the 3 different ciphertexts and read in 256 blocks of them at a time for decryption. Following the equation to solve for A, I used the values of ci I created and multiplied the corresponding ci to its 256 bit block, added them together and took the mod of M (the 3 n’s multiplied together). Solve p root was then called with

3 and a as its input parameters. The output block had its padding 0s shaved off and the combined decrypted text was written to the cracked file at the end of the process.