Modern Cryptography Research

Joseph Colton

September 2018

The Need to Understand

In my elementary years, I was first introduced to basic cryptography. Normally, I saw an alphabet replacement where each letter was replaced with a number. Often, you could figure out the key by solving a crossword puzzle or going through some other academic exercise.

This first introduction to cryptography interested me and soon my siblings and I were creating codes and encrypting messages for each other. There was, in us, the impression that the codes were unbreakable and messages could only be decrypted if one had the key. Mostly, the idea of breaking codes never really occurred to me.

During my undergraduate Computer Science education we were presented with a couple of different exercises where we had to break the Caesar Cipher encryption algorithm by shifting the characters one at a time and searching through a word list to find the most probable match. This approach seemed obvious at the time, but it was not until I reached graduate school that I started implementing algorithms to break that cipher using frequency analysis and other approaches.

It seemed to me that all exercises in encrypting and decrypting messages in the academic setting used simple algorithms and easy to implement schemes. The commercial world had abandoned these algorithms decades ago in favor of more complex modern algorithms such as the Data Encryption Standard (DES), triple DES, and eventually Advanced Encryption Standard (AES). In order to solve key problems they had embraced the Diffie-Hellman algorithm and RSA.

In the computer field we are regularly telling students about these algorithms and explaining the pros and cons of using each of them and what purpose they are there for. Just like a car salesman, we were regurgitating sales information and talking points without ever having built a car ourselves. Sure, I had seen write-ups on the inner workings of the algorithms, but I had never really written them and seen what they really do. I did not really know the issues and strengths of the algorithms.

The curiosity and need to understand ate at me, but in the IT world, you have to make decisions about which things you need to know and which you will just accept at face value. There is a time commitment to learning anything new, and we are constantly learning new things.

This sabbatical provided an opportunity to take the time to implement the algorithms and try to really understand how these modern cryptographic methods are superior to the older methods of encrypting data. I decided to focus on Data Encryption Standard (DES), Triple Data Encryption Algorithm/Triple DES (TDEA/3DES), Advanced Encryption Standard (AES), and RSA because these are some of the most well known algorithms.

DES and 3DES are algorithms which are now listed as obsolete because they are now “broken” and AES and RSA are still used in normal everyday commercial cryptography. I figured by studying right at the break point, I could see the difference between the broken and the operating algorithms. This would provide an opportunity to learn and really understand.

From Documents to Code

While I have a masters degree in computers, I think most of my abilities in programming came from my undergraduate program. I have often claimed that my biggest achievement in my masters program, other than getting the piece of paper, is that I learned how to read an academic paper without falling asleep after every paragraph. This sabbatical has given me an opportunity to show myself that I really learned that skill and that I can make sense of complex academic papers.

My first task was to obtain the original or followup documents that define and describe the algorithms used to encrypt and decrypt the data. Through Wikipedia articles I was able to learn the names of the official documents and was then able to search for the documents. When I had the documents, I started reading them and implementing each step of the algorithms.

Since these algorithms work at the bit and byte level, I decided I would code the algorithms in a language that operated well at that level. I decided to program in the C programming language since that is about as close to machine code as you can get and still be in a higher programming language.

Almost immediately, I ran into problems. The words I was reading appeared to be English, but I was not understanding. I was often reminded of a scene in the movie, "Three Amigos" when one character asks another, "Would you say I have a plethora of pinatas?" Plethora? Why? The use of difficult words in this case and in the encryption documentation appears to show literary self-superiority, nothing more.

The next major obstacle I ran into was the complexity of performing the bit wise operations. Normally, when programmers are doing code, they are operating at the character or byte level. When you have to perform operations on individual bits, you have to find a way to separate those bits and later recombine them together.

DES was quite annoying in that it used structures of different sizes. You had to use 64 bits, 56 bits, 48 bits, 32 bits, 28 bits, 6 bits, and 4 bits for different parts of the algorithm. You also had to regularly convert between different bit structure sizes. I was regularly having problems, and then you had to keep track of each individual bit.

Eventually, I was able to implement the DES algorithm. The algorithm was complex, but it was comprised of many iterations of the same basic steps. The algorithm used permutation of initial bits, then contained multiple rounds of using the XOR operation on the data using some data generated from the key using shifts and substitutions depending on the round number we were on.

AES was able to solve the problem of strange bit sized structures and was better able to standardize. This was not to secure the algorithm, but to make it easier to perform fast calculations using older hardware. AES was intended to be both secure and fast.

I assumed the longer key sizes of 128 bits, 192 bits, and 256 bits would make the algorithm a lot more secure, but in looking at the message that goes in and the encrypted text I discovered the data size was the same even with larger keys. This meant that frequency analysis attacks would be just as effective with larger keys since the block size never changes.

Additionally, I had assumed the AES algorithm would employ much more complex encryption techniques than DES, but I was wrong. The AES algorithm uses the same shifts, substitutions, and XOR operations. All of these are just wrapped up in multiple rounds of processing.

My investigation into DES, 3DES, and AES revealed that the algorithms were very similar and no fundamentally different operations were happening. Basically, AES is just DES simplified, remixed, and expanded to include a larger message and key.

The RSA algorithm is where the public key and private key pairs come from in asymmetric encryption. The algorithm was a little complex, but easier to implement than AES algorithm, at least for small prime numbers. If I used larger prime numbers I would have to use much more complex coding practices to handle large numbers. As it was, I still had to write functions to raise large numbers to larger prime numbers and then apply the modulus operation. This was tough.

I discovered that RSA has a weakness to multiple prime numbers, in that if you pick some prime numbers for your calculations, they generate keys that do not work. Here the public and private key selection needs to be randomized and then verified to make sure the keys actually work. The high volume of failed keys in the lower prime numbers was a bit annoying, and made randomized testing a bit harder.

I collected the parts of my project and put the code on the citstudent server at the following URL: http://citstudent.lanecc.net/~joseph/ColtonCrypt.zip

There you can see the code and see how it actually operates.

Changes

In addition to writing the software to implement the above listed algorithms, I wanted to also write ElGamal and Diffie-Hellman. I also wanted to go to the DEFCON Security Conference in late July to learn more about current research directions.

After working on this material for a while, and decided that I needed to update some of my industry certifications that I had let lapse a decade earlier. I studied the Cisco Certified Network Associate materials and went and took my certification exams and got recertified. While this does not directly play into the cryptography research, it was personal growth and something I have wanted to do for a long time, but just never found the time.

My renewed certifications will now make it possible for me to teach the Cisco curriculum, which will in turn make it easier for my students to get jobs working in networking.

The DEFCON security conference plans were for after the official ending of my sabbatical, but were put on hold when the governor, the college president, my dean, and outside organizations decided to try the Apprenti program here at Lane as an alternative method to get students into employment. I ended up taking a 3 week teaching position which got in the way of my plans. I do however, plan to try to go to the conference next year.

Final Thoughts

Anyone who wanted to actually use these algorithms should use publicly available open source libraries that implement the protocols and have been tested and troubleshot. So, what advantage have I provided by doing this research?

In doing this research, I have learned about the algorithms and now know what they are and what they are not. I have met that internal desire to know how the algorithms work and their real benefits and advantages over other algorithms.

Since we, in my division, are looking into creating security degrees or certificates, it helps when at least one of the faculty actually understands how the algorithms work and what the real weaknesses are. This information can then be passed along to students and will help them in their education.

This could provide potential future expansion as we look into additional classes or content to add to existing curriculum. The college could potentially benefit from an additional attracting exercise or two we could add that would help bring students interested in learning more about the computer security world.

Reference Material

- Wikipedia: Data Encryption Standard – https://en.wikipedia.org/wiki/Data_Encryption_Standard
- Federal Information Processing Standard Publication 46-3: Data Encryption Standard – https://csrc.nist.gov/csrc/media/publications/fips/46/3/archive/1999-10-25/documents/fips46-3.pdf
- Wikipedia: Advanced Encryption Standard – https://en.wikipedia.org/wiki/Advanced_Encryption_Standard
- Federal Information Processing Standard Publication 197: Advanced Encryption Standard – https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.197.pdf
- Wikipedia: Diffie-Hellman key exchange – https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange
- Wikipedia: RSA (cryptosystem) - https://en.wikipedia.org/wiki/RSA_(cryptosystem)