TECHNIQUES OF ASYMMETRIC FILE ENCRYPTION

Computer Systems Research Alvin Li Class of 2005

Abstract

As more and more people are linking to the Internet, threats to the security and privacy of information continue to grow. To solve this growing problem, encryption programs have been created to protect privacy during a transfer of files and to make sure that sensitive files will be protected. My project is to create an asymmetric file encryption program. This means that encrypted files will need a pass-key to open that will be different from the key used to encrypt. This program could be applied practically to protect files during transfers.

Background and Research

This area of research has been extensively researched in the past. Even before the widespread use of the Internet, file encryption had been developed for many years. At the moment, there exists numerous programs capable of encrypting files on a large scale. Some of these are very efficient, able to not only encrypt, but also compress files. There are many kinds of encryption. Two are public-key encryption and private-key encryption. This project will focus on private-key encryption methods. There are also many different types of encryption methods, including translation tables, data repositioning, and word/byte rotation. I have not yet decided which method to use. In the early stages of the project, I will use a downloaded encryption algorithm in my program.
When the other aspects of the program are working, I will write my own encryption algorithm. This way, I will utilize and build
off of existing work in this area.

Procedure and Development

Currently, I have completed the first version of my program. I have used a basic RSA algorithm as the basis for my project. The theory behind RSA is simple, but the implementation is difficult.

RSA was invented in 1977. It uses the fact that products of large primes are very hard to factor to its full advantage. Lets say you have two large primes, p and q. The public key would be the number p * q. This key is used to encrypt the message. The private key, would be either p or q. It is near impossible to derive p and q from p * q because factoring p * q is very difficult. If p* q is a 128 bits large, then you would need to try dividing p * q by 2^64 numbers to find p or q. This is because you need to try at most root n numbers to factor n. Naturally, trying to solve the cipher by brute force would take practically forever. This is what makes RSA so strong, the fact that it is pretty much unbreakable. With the private key p, you can decrypt the original message. The operations below shows how this is done.
public key n = p * q
private key large primes p and q
Let e be a random encryption exponent that is less than n and has no factors in common with (p - 1) or (q - 1)
Calculate the decryption exponent d which satisfies e * d mod (p - 1) * (q - 1) = 1
Then, the encryption function is E(m) = m ^ e mod n, for any message m
The decryption function is D(c) = c ^ d mod n, for any ciphertext c

This is the basic procedure for my RSA encryption program.

Results

Let me discuss the results of my project. The cryptext is very random, with ASCII values approximately randomly distributed throughout the encrypted text. I came to this conclusion by running tests on the encrypted data.
Also, the encrypted text is completely patternless, which is another essential component. Frequency analysis is a form of cryptoanalysis. Because the frequency of some characters in the English language occur more often than others (such as the letter 'e' and 's'),