Understanding Encryption and Password Protection

By Kevin Day, Chief Security Officer
Attach Plus, LLC
www.attachplus.com

Encryption is critically important to any sort of security because a file that only has a password on it and is not encrypted can be opened and read by anyone with the right skills. That is worse than no security at all, because it gives a false impression of security. When you apply a password to a file in Attach Plus, the contents of the file are actually encrypted. Without the password, the decryption keys are not known, so the content cannot be recovered by an attacker.

There are several key concepts to understand regarding encryption:

1. The two classes of encryption algorithms
2. Strategies for key exchange
3. How encryption and decryption keys are generated in symmetric algorithms
4. The importance of using good password

Two Classes of Encryption Algorithms

Encryption algorithms take a plain text stream of data and an encryption key and generate a cipher text stream of data. There are two broad classifications of encryption algorithms, split by whether they use the same key for encryption as for decryption

1. Symmetric – this means that the same encryption key is used for decryption
2. Asymmetric – this means that there are *different* keys for encryption than for decryption

.

In general, symmetric encryption algorithms are fast. Asymmetric algorithms are very slow. This has no bearing on the relative *security* of either class of algorithms - just the speed with which the algorithm can be executed.

The most widely used symmetric encryption algorithms are 3DES and AES. These use shared keys, and are actually the algorithms responsible for the vast bulk of data transferred securely over the Internet.

The most widely used Asymmetric encryption algorithm is referred to as 'Public Key'. The idea here is pretty interesting - your recipient can actually publish their **en**cryption key, and anyone can use that key to encrypt data. But only the recipient knows the private **de**cryption key. This sounds ideal, except for two glaring limitations:

1. Asymmetric encryption is slow. This issue, as it turns out, is easy to fix. You use the recipient's public key and asymmetric encryption to encrypt a randomly generated *symmetric* key. You send the encrypted package to the recipient. The recipient is able to decrypt the package (because they have the private key), and obtain the symmetric key. You then do all of your exchange using symmetric algorithms, which are much faster. This strategy is used quite frequently – in fact, most secure exchanges of data over the Internet begin with a single asymmetric exchange to transfer temporary keys, followed by the symmetric exchange of the actual data.

But there's a big remaining problem:

2. The recipient has to have a public key. In a situation where two large businesses are exchanging data (think of an ATM communicating with the bank's servers), the cost of setting up what is referred to as 'Public Key Infrastructure' (or PKI) is not a deterrent. They pay money to have certificates generated and authenticated (authentication is an entirely different aspect of asymmetric encryption). They set up servers to publish these keys, etc.

   With smaller businesses, setting up public key certificates is still within reach (most firms have some sort of certificate for use with their website), but it does add a lot of complexity. The security of the private key must also be managed. Compromising a private key is horrendous from a security perspective. Making sure that you don't accidentally write a private key to a disk somewhere is incredibly important. In fact, most schemes involve encrypting the private key with – that's right – a password using symmetric encryption.

   When you are dealing with end user recipients, this problem becomes untenable. It would be convenient (though a bit intrusive) if every American was issued a public/private key at birth, and it was maintained in a central repository. But that just isn't the case. There are services that will issue public/private keys, but this is a huge technological barrier to entry for the vast majority of people you might want to communicate with.

Key Exchange

Exchange of encryption and/or decryption keys is probably one of the most difficult aspects of cryptography. At a purely technical level, symmetric/asymmetric hybrid encryption schemes are appealing from a key exchange perspective, but they have real world limitations that make them unrealistic for use in business to customer communication.

There are also distinct advantages to pure symmetric encryption. Specifically, the question of key exchange is completely deferred. It takes place with a phone call, or face-to-face meeting. Central repositories of keys don't need to be created, managed, paid for, etc. The end recipient is probably going to have to use a password to protect their private keys regardless, so from their perspective, they still have to type the password in.

The question of key exchange is actually where the Attach Plus design began. We wanted to provide a solution that was incredibly simple for the sender and the recipient. We needed this to be part of the process of actually attaching files to emails (trying to get people to context switch to handle encryption was laughable). In addition, we needed the application to 'just work', regardless of whom your recipient is – and preferably without the recipient having to install anything on their computers, or sign up for any special accounts, etc...

Encryption Strength and How Keys are Generated in Symmetric Algorithms

No discussion of encryption would be complete without talking about encryption strength. Interestingly, a well designed encryption algorithm is, itself, un-crackable without the decryption key. But we can guess the decryption key and just try them all, right? So the best encryption algorithm on the planet is useless if it is protected by a 3 digit number - a computer can try 999 different combinations very quickly. So once we've cleared the hurdle that a particular algorithm is well designed, we have to ensure that the size of the encryption key is big enough that it would take an absurdly long time for an attacker to guess every single combination.

The size of the encryption key is usually measured in 'bits'. Most encrypted traffic that goes over the Internet uses 128-bit encryption. Generally speaking, the larger the key, the stronger the encryption - but you have to be a bit careful with this. A modern computer would require many, many thousands of years to try every combination of a 128-bit encryption key (this is called a 'brute force' attack). By going to 256-bit encryption, you can change that to millions and millions

of years. But the *effective* security is the same. 128-bit key lengths have been found by the security community to provide a nice balance of encryption that would take a really, really long time to crack with brute force techniques, without the key becoming ridiculously long.

When you supply a password to Attach Plus, the software takes the password and turns it into a 128 (or 256) bit key using an algorithm called a one-way hashing algorithm. If you input the same password, you get the same 128 or 256-bits of effectively random data. But it is not possible to start with the 128-bits of data and obtain the password.

The files generated by Attach Plus are actually not password protected. In fact, the password isn't part of the file at all. Instead, the contents are encrypted using a symmetric algorithm. The key for the symmetric algorithm is obtained by taking the password and processing it with a one-way hashing algorithm.

When a user opens the attachment, they are prompted for a password. The password is not compared against any value in the attachment. The password is used to mathematically obtain the decryption key.

The Importance of Using Good Passwords

Let's start with examples of bad passwords:

1. Regular English words (i.e. not including numbers, symbols, combination of upper and lower case)
2. A client's social security number, or any other piece of information that might be guessable
3. Any password that you e-mail to the recipient (if an attacker has access to one email, they probably have access to all of them!)
4. Anything that is short (shorter than 8 characters is generally considered to be bad)

My recommendation is to let the recipient decide the password. That puts the onus on them to choose something that is secure.

By analogy: the strongest safe in the world does no good if the combination is written on the door.

Happy encrypting!