

Foundations of Cryptography

by: LearnSecurityOnline, 08/17/2005

<http://www.securitydocs.com/library/3547>

Cryptography has been employed for keeping secrets since the time of Caesar. From the simplest ciphers of shifting letters, to mathematically provably secure ciphers of today, cryptography has progressed a long way. It also has widened to a number of uses and capabilities to fit an ever growing number of applications. Cryptography makes it possible to keep data secure over an insecure network. It also makes it possible to keep private data on your computer safe from prying eyes. Even car thieves can be foiled by crypto systems in your remote unlock system.

The basic idea of cryptography is to take a plaintext message, combine it with a key, and get ciphertext output. Once ciphertext is generated, its secrecy is not that important as long as the key is secret. Only those with the key to decrypt the message are able to read it. The process of encrypting plaintext messages is encryption. Getting the plaintext back from the ciphertext is decryption. The process of trying to break a cryptosystem is cryptanalysis.

An introduction to cryptography is presented by the SSH.com group at <http://www.ssh.fi/support/cryptography/introduction/>.

Foundations of Cryptography

The earliest use of cryptography was used by Caesar to transmit vital commands to and from his officers. The method he used was simple, but highly effective for the time. The idea was to take each letter, and shift it by a number of characters. So for a shift of 2, A would become C, E would become G, M would become O, and so on. What made this method even better was that it was simple enough to be done while writing, and it is even possible to train yourself to do it in your head.

Another variant of the same scheme is just to randomly replace one letter for another, or the transposition cipher. The advantage here is that instead of 25 possible ways to permute the text, there are 25!. As anyone who has worked the letter permutation puzzles in newspapers can cite, although mathematically they look intractable, they are very easy to solve. The primary method used to transposition ciphers on language text is to use a frequency table. The idea is that in language certain characters appear at some frequency. By analyzing the frequency of characters in the ciphertext, a best match can be made that minimizes the difference between the frequencies of the ciphertext, and the frequency normal of the language. With a small amount of correction by looking at the results and giving feedback, these ciphers hold no problem for the cracker.

A simple modification of the transposition cipher was the invention of Blaise de Vigenere and the use of a key. The problem with the transposition cipher was that it had the easily exploited frequency analysis problem. With a Vigenere cipher, a block of transposition tables is formed. The key determines which table is used. Whenever letters of the key used to select the table run out, it is reused from the start. This means that for any letter, depending on its position, it could translate to any number of characters in the ciphertext.

Although the analysis seems obvious, split the cipher text into n frequency analysis problems, where n is the length of the key, it took some time to get to this. For several hundred years no one thought up of this, or a way to extract the key length. In 1863, Kasiski proposed a technique for discovering the key length, and then doing the decomposition of the problem into n frequency attacks. His trick for discovering the length of the key was the fact that if an identical portion of the plaintext lined up in the same way with the key as it previously had, it would produce an identical portion of the ciphertext. By observing these identical portions of ciphertexts and measuring their distance it is possible to come up with the most common divisible factors into these distances. One of the most common divisible factors will be the key length.

The beginning of modern cryptography is the use of the XOR digital operation. XOR is an invertible bit level manipulation. It is defined by the following table.

Input1	Input2	Output
0	0	0

```

0   1   1
1   0   1
1   1   0

```

The most simple crypto system that comes out of the XOR operation, is using the bit representation of a plaintext, and XORing it with a key. The most obvious use of the key is to just repeat it whenever bits of key run out. This however has the same problems that lead to a very easy statistical attack on it. On the other end though, if a key of length equal to the message you want to encrypt is used and securely transmitted, then the encrypted message is perfectly secure, as long as the key is not repeated and not compromised. The use of a very long key in a set of repeated XOR operations is called a One Time Pad (OTP). The problem with this method is securely exchanging keys when you need to encrypt data.

A very complete introduction to historical ciphers and work to date can be found at <http://starbase.trincoll.edu/~crypto/historical/>.

Encryption Modes and Techniques

The two principle modes of operation for ciphers are stream and block ciphers. Stream ciphers are the most intuitive. You put in a bit, and a bit comes out. Incoming ciphertext should ideally produce what appears to be random output. Although stream ciphers have their uses, the more common mode of operation is the block cipher. In this mode a block of bits is operated on as a whole and results in a block of ciphertext. Excess space in the block is padded with random data to ensure proper functioning of the cipher. The most famous block cipher is DES, or the Data Encryption Standard. Other common ones are Blowfish, IDEA, and the new AES.

ECB, CBC, and CTR modes

Block ciphers also have a number of ways of operating to prevent a replay of plaintext from producing an identical output in the ciphertext. The most obvious way to use a block cipher is to fill the block with plaintext, and then to run the cipher on it. When using this method, Electronic Codebook (ECB), two blocks that are identical will encrypt to the same ciphertext. To solve this undesirable property, two other modes are used instead. The most common one, because of standardization is the Cipher Block Chaining, or CBC. In this process the previous ciphertext is XOR'ed with the plaintext block before it is encrypted. The most commonly cited problem for this mode of operation is that if a block is lost then the encryption and decryption are out of synchronization. The other common way to operate is to add a counter into each block that counts in a specified manner that should be reasonably unpredictable for someone trying to break the crypto-system.

A flow of ECB, CBC, and CTR modes

```
ECB: Plaintext-> |Encryptor| -> Ciphertext
```

```
CBC: Plaintext  -> |XOR| -> |Encryptor| -> Ciphertext
     Past block ---^
```

```
CTR: Plaintext  -> |Combine| -> |Encryptor| -> Ciphertext
     Counter  -----^
```

Fundamental Operations

Along with the listed modes of operation, there are also some useful techniques that are frequently used in cryptography. The first one was already mentioned before, the XOR function. XOR is frequently used in combining two inputs to form an output.

The idea is that knowing only one part of the three reveals no information about the other two, and is the simplest function to have this property on a binary level.

The lookup table has also provided a key building block of cryptography systems. The general idea is that n bits of input are used to lookup m bits of output. The inputs are typically some function of the key and plaintext. Outputs may be chained through a number of these lookup tables. The most famous lookup tables are the S-Boxes of DES. With the amount of research into the functioning of DES's S-Boxes has shown how good tables are formed, and how to attack poorly created ones. For DES, the key to its security is in the mathematical security of its S-Boxes.

The last important tool we will mention here for encryption algorithms is the concept of permutation. Although not actually secure itself, it adds a layer of random distribution of data to input for other functions that need a more even distribution of data to be secure. Common operations here usually involve swapping bytes, shifting to the left or right, or shuffling the entire bit-space.

A description of common modes for encryption algorithms is located at <http://crypto-systems.com/modes.html>

Key Distribution

The problem with the crypto-systems of past was keeping the key secure, and securely communicating the key to the other party. The general problem of this is key distribution. In past the idea was to arrange a secure meeting and exchange keys for later use. For many uses, this is really not practical. During World War II, German submarines had to go for months without docking, so their Enigma codebooks were valid for very long periods of time. This also presented the problem that if a codebook was captured, that all communications past, present, and future would be compromised. This problem remained unsolved until the invention of the RSA encryption system by Rivest, Shamir and, Adleman.

Public key has solved the problem of key distribution. In a public key cryptography system, there are two keys. One is for encrypting, and the other is for decrypting. Divulging one key does not give the attacker any useful information about the other. The mathematical grounds for this is that there are certain problems that are very easy to construct, but are extremely difficult to solve. There are a number of these problems, but the one that RSA uses is factoring. Given existing prime generator functions, it is easy to generate arbitrarily large prime numbers. Whenever the product of these two prime numbers is taken, it becomes a near impossible task to discover the original numbers. The step taken on this is that Rivest, Shamir, and Adleman found a way to derive three other numbers from the two primes and their product. A property was derived that two of the numbers could be revealed for use in an equation of modular arithmetic, but the equation could not be solved in the reverse manner unless the other pair was used in the equation. With this public key cryptography was born.

Key services

The only problem with public key cryptography is ensuring that the person who is giving you their key, is really them, and it is their key. The man in the middle attack is a present danger to public key cryptography, where a person in the middle of two parties wishing to communicate pretends to be the endpoint to each party. The solution to this is a system where a trusted party will authenticate that a person and their public key are really supposed to be paired. In the end the security comes down to the ability of the third party in its authentication role and its trustworthiness.

Another interesting aspect of key management is the use of escrow services. Once proposed by the NSA with its Clipper encryption system, the idea is that a back-door key can be sent to agencies who can reveal the key to investigative authorities if they need to monitor the encryption system. Other uses could be for security against lost keys and the resulting loss of data. There are many aspects to the idea of key escrow, and there are very strong feelings about the use of escrow on both sides. On one hand, it provides a way to decrypt data when a key is lost, or investigative personnel need to monitor communications. On the other, it provides a way to break the cryptography by obtaining the key from the escrow agents.

A detailed, although technical, discussion of key distribution is at http://www.cs.utsa.edu/~wagner/laws/public_dist.html.

Uses for Cryptosystems

Other than the obvious use of keeping messages secure, there are a number of technologies that can be built on top of a secure transport protocol. One of the most commonly used is the hash function. The idea is that a small message can be created to determine if a message is the same as the one that was used to generate the hash. If two sides securely transmit the secure hash, then they can verify if they have the same message. This is message verification. The most common message hash algorithm is MD5, and is used for many distributors of programs as their hash algorithm of choice for their packages.

Message Signing and its uses

Another use which is very closely related to verification is signing. Here a message of a secure hash and an identifier are combined into a publicly encrypted message where only one party has the encryption key, and everyone else is able to decrypt it. The idea here is if a party wants to ensure that it can distribute messages with everyone able to determine that it was the one who sent it, they can distribute the decrypt keys to everyone. Whenever a message and signature are received, the signature is decrypted and if the message hash matches and the signer's identification appears with the rest of the decryption, then the message really came from them.

A future use of signing is already in development and will help secure computers from harmful code. The idea is that code will be signed by the vendors who create it. Whenever code is run, it must have been signed by a trusted vendor or the system will not execute it. This provides users with the ability to choose whose code they run, and verify the code is actually from that source. Some people fear that it could be used to lock some software distributors out of potential customers systems.

Some people are even working on integrating this level of cryptography into the lowest software level of data storage, the file-system. With integrated cryptography, hashing, and signing files would be more secure, authenticate who created them, and have greater privacy. The greatest backers of this sort of technology are the people who have a vested interest in data, the software and music industries.

Although public key cryptography enables a number of very exciting technologies, it also has a major drawback of speed. For speed the secret key systems can not be beat. This results from the mathematical complexity that is used to secure the data in public key crypto-systems. The solution used by many to this problem is to simply use public key cryptography to exchange a key for use in private key cryptography. An added benefit here is that a randomly generated key can be used for each session, and that a compromise of one session does not necessarily lead to a compromise of all sessions. After the two sides are done exchanging keys, they are able to run at the full speed of secret key cryptography systems.

A survey of uses people have for cryptography is here <http://www.viacorp.com/crypto.html>.

Cryptography Products

Arguably the most common cryptography product is the secure https protocol. It is the foundation of most secure interactions on the Internet, and keeps secure the vast majority of electronic commerce. Almost every machine that has a web browser has support for https. The main challenge with https for servers is the initial setup, which uses costly public key algorithms. To offload this computation, an administrator can purchase hardware accelerator cards for SSL transactions.

The second most common cryptography product has to be the signing systems in operating systems. Popularized by Microsoft's Windows Update webpage, the ability of Operating Systems to authenticate their updates is a major step in distributing updates over the Internet. Further support is even being added so that you can trust code from certain vendors, and if anyone tried to send you other code, or send you modified code, that your system could prevent it from running unless you explicitly told it to do so.

In line for a third is the widely used Secure Shell client and server system. The idea of Secure Shell, or SSH, is to provide secure remote terminal access to a system. Not so long ago when large Unix systems dominated the server market and networks were private, un-encrypted protocols for obtaining a console on a Unix machine were used. The most common protocol was telnet, which gave the user what appeared to be a terminal, but over the network. Other insecure programs such as remote copy (rcp), file transfer protocol (ftp), and remote shell (rsh) were replaced by alternatives secured with SSH. The most popular SSH distribution is OpenSSH, available at <http://www.openssh.org/>.

Public key cryptography has not completely destroyed the use of private key systems in widespread use. One such product that came out of MIT is called Kerberos. It is a system of centrally stored credentials, and a network of machines that accept a list of credentials listed in the central machine. The primary use of Kerberos is in a system where untrusted users are in a network of trusted machines. To access a resource, a client uses their ticket granting ticket to get a ticket from the central key domain controller for a resource it wishes to user.

Among many there has been the idea of encrypting data at a lower layer. Most applications should not be bothered by the details of implementing a secure crypto-system. Usually people not experienced in developing such systems will build them with many flaws in the implementation. To ease the deployment of cryptography, and ensure secure implementations, a standard of cryptography has been proposed as an extension to the Internet Protocol. IPSEC provides all the aspects of cryptography that HTTPS and SSH have, but at a lower level. For computer systems targeted to business customers, IPSEC acceleration in the network card is a standard feature.

Of personal cryptography products, the best known is the PGP suite. PGP provides all of the useful functions of cryptography that people would use in everyday computing. The distributions of PGP also integrate it very well into programs such as e-mail clients, system tools, and other programs. From the start of installation, it walks the user through creating a public and private key pair, uploading it to a key-server, and using the basic functions of encryption, decryption, and signing in their applications. PGP can be found at <http://www.pgp.com/>.

Conclusion

Cryptography is a very active field of research. Some of the essential foundations of cryptography are still unproven, such as if P space complexity algorithms are equivalent to NP space complexity algorithms. For someone wishing to investigate the actual use of cryptography further, a strong background in math is a necessity. Other parts of the cryptography field do not need such focus in math. There are many open areas of implementation and integration with other products. One thing is ensured though, and that is we will see an ever growing presence of cryptography in our lives.