Symmetric and Asymmetric Algorithms

Telle Miller

CSIS 463-B01: Course Project 1

Liberty University

**Introduction**

Encryption and decryption are both integral parts of secure online interaction. Cryptography, or protecting information through codes, leverages these processes to allow users to safely interact amongst themselves. The encryption and decryption processes are used nearly everywhere online to protect user traffic and maintain anonymity. A key part of these processes is algorithms. Algorithms, or a process to be followed in data transformation, define how data is encrypted and decrypted quickly and effectively. These algorithms vary widely, but the result is data that is obfuscated and hopefully protected from prying eyes.

**Algorithms**

Algorithms are mathematical formulas that have been developed to transform data, called plaintext, into a more cryptic form, called ciphertext. The efficiency, both in terms of speed and level of encryption, vary between the differing algorithms. These cryptographic algorithms use variables, referred to as keys, to vary the output of the algorithm and determine the level of security provided. These keys introduce the two types of encryption used in modern times; symmetric and asymmetric. These two categories of cryptographic systems have differing use cases, speeds, methods, advantages, and disadvantages.

**Keys**

As previously mentioned, keys are variables introduced into a cryptographic algorithm to define the effectiveness of the encryption sequence. Keys can be a variety of lengths, with shorter keys typically being weaker and longer keys being more secure albeit with quickly diminishing returns. Modern cryptography is largely locked to the digital world, meaning that keys are ultimately represented in bits. This means that the number of possible keys is a constant relationship with the number of bits used for a key. This allows for the defining of a formula to determine the total possible keys, shown below.

Number of possible keys → , where X is key length in bits.

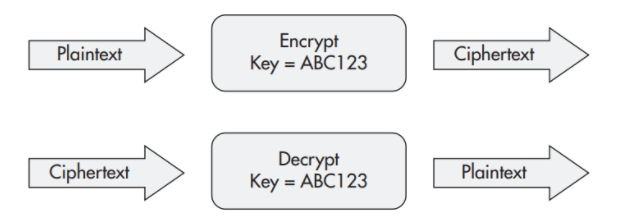
For example, a key with a length of 8 bits would have 256 possible combinations.

|  |  |
| --- | --- |
| **Length in bits** | **Possible keys** |
| 1 | 0, 1 |
| 2 | 00, 01, 10, 11 |
| 3 | 000, 001, 010, 011, 100, 101, 110, 111 |

The totality of possible keys allowed by a given algorithm is referred to as the keyspace. The length and possible keys is an important feature of an algorithm, as an increase in length provides greater protection against brute forcing. Brute forcing attacks are attempts to decrypt a file by using every key possible. This method of attack requires immense computing power, but as computing powers increase with time, the bit length of keys is continually required to increase. Previously secure algorithms that used 56-bit keys are trivial for some entities to crack in modern times. Most modern algorithms offer 128 and 256-bit keys, as the time to crack these codes is practically insurmountable.

**Symmetric**

Symmetric encryption algorithms are the oldest forms of encryption algorithms, with uses stretching back thousands of years. The symmetric encryption algorithms are sometimes referred to as classic, traditional, or conventional encryption (Oriyano, 2018). Symmetric encryption gets its name from the way it processes data from plaintext to ciphertext. Symmetric algorithms use the same key for both encryption and decryption. The use of this key on both sides of the algorithm is the cause of the “symmetry” of this system.

Example of symmetric encryption

**Advantages**

Symmetric algorithms have varying strengths and weaknesses in the modern world. Some of the advantages of symmetric algorithms is their ability to preserve confidentiality, speed, simplicity, and proving authenticity (Roeder).

**Disadvantages**

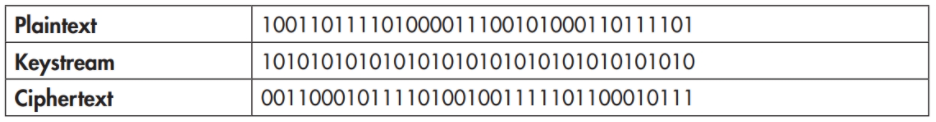
Some disadvantages caused by symmetric algorithms are key management and non-repudiation. Non-repudiation is the “assurance that the sender of information is provided with proof of delivery and the recipient is provided with proof of the sender’s identity, so neither can later deny having processed the information” (Oriyano, 2018). Key management is the requirement to handle keys in a secure and practical way so that the encryption remains unbroken, and decryption remains accessible. One of the major problems with symmetric encryption is the hand off of the key between sender and recipient. Once a file is encrypted with a key, this key must be given to the recipient so they can open the file upon receiving it. However, if the method of communication between the two is compromised, the attacker can easily capture this key and use it for any files that are traded between the sender and recipient. Throughout history, this problem was typically solved by physically handing off the key to the recipient so that the risk of interception was mitigated. However, as the Internet connects individuals that cannot physically interact easily, they are relegated to sending the keys “out-of-band”. This is the process of sending the key on a different channel than the encrypted file, in hopes that the separation of the two prevents attackers from getting both.

**Stream vs Block cipher**

Varying symmetric algorithms use different techniques to achieve their encryption. These techniques are split into two categories, blocks and streams. Block ciphers take the plaintext data and break it into pre-determined blocks of data and then operates on them individually. The size of these blocks varies between algorithms and are set during the design of the algorithm so that they are non-variable. If the data size is not divisible by the block size with a remainder of zero, the block algorithm will append extra bits to fill the last block (Cusick & Stanica, 2009). For example, if a particular algorithm uses block sizes of 64 bits, and the data to be encrypted is 92 bits, the algorithm will operate on two blocks. The first block will contain the first 64 bits of plaintext and the second block will contain the remaining 28 bits of plaintext with 36 bits of padding added to fill the block. Some of the advantages and disadvantages of using a block cipher are listed below.

|  |  |
| --- | --- |
| **Advantage** | **Disadvantage** |
| Block ciphers are easier to implement in software and hardware | Block ciphers can require more memory than stream ciphers |
| Block ciphers can provide integrity protection | Block ciphers are susceptible to data corruption |
| It’s beneficial to use a block cipher when the size of data to be encrypted is known ahead of time |  |

The other form of symmetric ciphers are stream ciphers. Stream ciphers are typically much faster in comparison to block ciphers and often require less memory. Stream ciphers perform encryption by working on each individual bit, one at a time, converting them from plaintext to ciphertext. This method is accomplished by utilizing a keystream, which randomly or pseudo randomly generates a stream of 1s and 0s based off of the inserted key. This string of 1s and 0s is then processed with the plaintext to be encrypted using an exclusive OR (XOR). This is a binary process in which two strings of bits are transformed into one string based on their contents. An example XOR process is shown below.



The XOR process looks at the first position of each string of bits, and if the bit combination contains exactly one 1, a 1 is then appended to the final string of bits, otherwise a 0 is appended. This process is repeated for every bit in the plaintext data stream until every position has been obfuscated. For decryption, the ciphered text and keystream can be processed using an XOR to obtain the original plaintext data. Stream ciphers are ideal for situations where the amount of data being encrypted is unknown (Oriyano, 2018).

Both block ciphers and stream ciphers have a place in modern cryptography. There are tradeoffs for each form of symmetric encryption, even though the most common symmetric algorithms use block ciphers. It is important to keep in mind the functional requirements of a project when choosing an algorithm, as this is greatest deciding factor in which is better suited.

**Examples**

There are many modern examples of symmetric algorithms, as they are widely used for a variety of systems. Some of the most popular symmetric algorithms are listed below.

* **ROT13** – Simple substitution cipher, akin to the Caesar cipher. It provides little to no cryptographic security but is often used for puzzle and word games.
* **Blowfish** – Symmetric encryption algorithm that utilizes a block cipher. This algorithm is full open-source and has a 64-bit block size. It supports key lengths between 32 and 448 bits.
* **RC2** – Symmetric key block cipher developed in 1987. It supports a 64-bit block size and a variable key size up to 40 bits. This algorithm is susceptible to brute force key attacks.
* **RC6** – Another symmetric key block cipher, derived from RC5 and the previous generations of RC algorithms. This algorithm was developed to meet AES standards and has a block size of 128 bits. RC6 supports common key sizes of 128, 192, and 256 bits. The max supported bit size is 2040.
* **DES** – Symmetric key block cipher algorithm with a key size of 56 bits. DES has a block size of 64 bits and is susceptible to brute force attacks due to key length restrictions imposed by the NSA during the time of this algorithm’s creation.
* **3DES** – Symmetric key block cipher algorithm that aimed to fix the issues created by the short key length of DES. This algorithm functions similarly to DES but encrypts the same block three times. The key length of this algorithm is 168 bits, which is equal to three 56-bit keys from DES. Despite the goal during creation of this algorithm, 3DES is still susceptible to brute force attacks and considered mostly unsecure by NIST.
* **Advanced Encryption Standard (AES)** – Symmetric key block cipher that supports key lengths of 128, 192, and 256 bits. With a block size of 128 bits, the US government has adopted AES as a main method of encryption, replacing the former algorithm of choice, DES.

**Asymmetric**

Asymmetric encryption algorithms are a relatively new development in cryptography, especially in comparison to symmetric algorithms. Asymmetric algorithms were developed in the 1960s and 1970s as corresponding advances in mathematics and technology needed to implement them were occurring (Oriyano, 2018). Asymmetric algorithms, unlike symmetric algorithms, use two separate keys for its encryption and decryption processes. Every user of this system has a public and private key that are mathematically linked. This mathematical linking is typically performed by using a complex computation that is not easy to reverse, such as multiplication of primes.

**Advantages**

One of the biggest pitfalls of symmetric encryption is key management. There is the hard problem of safely transferring keys between participants, but beyond this, the need for unique keys grows rapidly as we expand the number of users. Every person using a symmetric encryption system must have a unique key and any time someone sends them a file, they must have that user’s key also. The formula below outlines the number of keys needed to be generated for a given size of users.

= Keys

|  |  |
| --- | --- |
| User population | Total keys to be generated |
| 10 | 45 |
| 100 | 4,950 |
| 5,000 | 12,497,500 |

This bulk of keyspace quickly becomes cumbersome to manage. This is one of the key advantages and problems solved by asymmetric encryption. Every user’s public key is attached to them and available on demand to other users. This means that users do not need to ever interact physically or request keys from each other. Asymmetric encryption is a great invention to implement on large, interconnected systems, such as the Internet.

**Disadvantages**

The major drawback of asymmetric encryption systems is the reliance on advanced technology systems. The need for extremely large key generation and the basic structure of the algorithms makes this system magnitudes slower, even up to factors of a 1,000. Therefore, asymmetric systems are typically reserved for small amounts of data (Oriyano, 2018). These extreme computation times mean that asymmetric encryption is nearly impossible without the aid of computers and explains it’s more recent development in the cryptographic timeline.

**Examples**

* **RSA** – Public key cryptosystem that is widely used for secure data transmission across the Internet. For practical use, most RSA systems use key lengths ranging from 1024 to 4096 bits (IEEE 1363).
* **Diffie-Hellman** – This public key cryptosystem was one of the first devised and paved the way for future asymmetric key algorithms. This system was more of a theoretical plan for future systems to implement off of, such as RSA.
* **Elliptical Curve Cryptography (ECC)** – ECC is at the foreground of modern cryptography because it solves most of the known security vulnerabilities against asymmetric key systems. ECC allows for small keys compared to non-ECC systems, thanks to leveraging the algebraic structure of elliptical curves for computations.
* **Digital Signature Algorithm (DSA)** – Public key cryptosystem that was proposed and adopted by the National Institute of Standards and Technology. The most recent revision of DSA is largely used for digital signature standards.

**References**

Oriyano, S.-P. (2018). *Cryptography: Infosec pro guide.* New York: McGraw-Hill Education.

Delfs, Hans, and Helmut Knebl. *Introduction to Cryptography: Principles and Applications.* Springer, 2007.

Roeder, T. (n.d.). *Symmetric-Key Cryptography.* Symmetric-key cryptography. <http://www.cs.cornell.edu/courses/cs5430/2010sp/TL03.symmetric.html>.

Ferguson, Niels; Schneier, Bruce (2003). *Practical Cryptography*. Wiley. ISBN 0-471-22357-3.

IEEE 1363: *Standard Specifications for Public-Key Cryptography*

"Advanced Encryption Standard (AES)" (PDF). *Federal Information Processing Standards*. 26 November 2001. doi:10.6028/NIST.FIPS.197. 197.

Cusick, Thomas W.; Stanica, Pantelimon (2009). *Cryptographic Boolean functions and applications.* Academic Press. ISBN 9780123748904.