Symmetric-key algorithms[1] are algorithms for cryptography that use the same cryptographic keys for both encryption of plaintext and decryption of ciphertext. The keys may be identical or there may be a simple transformation to go between the two keys[*citation needed*]. The keys, in practice, represent a shared secret between two or more parties that can be used to maintain a private information link.[2] This requirement that both parties have access to the secret key is one of the main drawbacks of symmetric key encryption, in comparison to public-key encryption (also known as asymmetric key encryption).[3]

Types of symmetric-key algorithms[edit]

Symmetric-key encryption can use either stream ciphers or block ciphers.[4]

- Stream ciphers encrypt the digits (typically bytes) of a message one at a time.
- Block ciphers take a number of bits and encrypt them as a single unit, padding the plaintext so that it is a multiple of the block size. Blocks of 64 bits were commonly used. The Advanced Encryption Standard (AES) algorithm approved by NIST in December 2001, and the GCM block cipher mode of operation use 128-bit blocks.

Implementations[edit]

Examples of popular symmetric algorithms include Twofish, Serpent, AES (Rijndael), Blowfish, CAST5, Kuznyechik, RC4, 3DES, Skipjack, Safer+/++ (Bluetooth), and IDEA.[5][6]

Cryptographic primitives based on symmetric ciphers[edit]

Symmetric ciphers are commonly used to achieve other cryptographic primitives than just encryption.[*citation needed*]

Encrypting a message does not guarantee that this message is not changed while encrypted. Hence often a message authentication code is added to a ciphertext to ensure that changes to the ciphertext will be noted by the receiver. Message authentication codes can be constructed from symmetric ciphers (e.g. CBC-MAC).[*citation needed*]

However, symmetric ciphers cannot be used for non-repudiation purposes except by involving additional parties. See the ISO/IEC 13888-2 standard.[*citation needed*]

Another application is to build hash functions from block ciphers. See one-way compression function for descriptions of several such methods.[*citation needed*]

Construction of symmetric ciphers[edit]

Many modern block ciphers are based on a construction proposed by Horst Feistel. Feistel's construction makes it possible to build invertible functions from other functions that are themselves not invertible.[*citation needed*]

Security of symmetric ciphers[edit]

Symmetric ciphers have historically been susceptible to known-plaintext attacks, chosen-plaintext attacks, differential cryptanalysis and linear cryptanalysis. Careful construction of the functions for each round can greatly reduce the chances of a successful attack.[*citation needed*]

Key management[edit]

Key establishment[edit]

Symmetric-key algorithms require both the sender and the recipient of a message to have the same secret key. All early cryptographic systems required one of those people to somehow receive a copy of that secret key over a physically secure channel.

Nearly all modern cryptographic systems still use symmetric-key algorithms internally to encrypt the bulk of the messages, but they eliminate the need for a physically secure channel by using Diffie–Hellman key exchange or some other public-key protocol to securely come to agreement on a fresh new secret key for each message (forward secrecy).

Key generation[edit]

When used with asymmetric ciphers for key transfer, pseudorandom key generators are nearly always used to generate the symmetric cipher session keys. However, lack of randomness in those generators or in their

initialization vectors is disastrous and has led to cryptanalytic breaks in the past. Therefore, it is essential that an implementation uses a source of high entropy for its initialization.[7][8][9]

Reciprocal cipher[edit]

A reciprocal cipher is a cipher where, just as one enters the plaintext into the cryptography system to get the ciphertext, one could enter the ciphertext into the same place in the system to get the plaintext. A reciprocal cipher is also sometimes referred as self-reciprocal cipher. Examples of reciprocal ciphers include:

Encryption algorithm example[edit]

The following steps should be followed to develop an encrypted text: [10]

- 1) Generate the ASCII value of the letters
- 2) Generate the corresponding binary value of it (binary value should be 8 digits e.g. for decimal 32 binary number should be 00100000)
- 3) Reverse the 8 digit's binary number
- 4) Take a 4 digits divisor (>=1000) as the Key
- 5) Divide the reversed number with the divisor
- 6) Store the remainder in first 3 digits & quotient in next 5 digits (remainder and quotient wouldn't be more than 3 digits and 5 digits long respectively. If any of these are less than 3 and 5 digits respectively we need to add required number of 0s (zeros) on the left hand side. So, this would be the ciphertext i.e. encrypted text.

Now store the remainder in first 3 digits & quotient in next 5 digits.

Let us see an example to apply the above mentioned steps:

Let, the character is "T". Now according to the steps, we will get the following:

- Step 1: ASCII of "T" is 84 in decimal.
- Step 2: The Binary value of 84 is 1010100. Since it is not an 8-bit binary number we need to make it an 8-bit number as per the encryption algorithm. So it would be 01010100
- Step 3: Reverse of this binary number would be 00101010
- Step 4: Let 1000 as divisor i.e. Key
- Step 5: Divide 00101010 (dividend) by 1000 (divisor)
- Step 6: The remainder would be 10 and the quotient would be 101. So as per the algorithm, the ciphertext would be 01000101 which is ASCII 69 in decimal i.e. "E"

5.3 Decryption algorithm

- Step 1: Multiply last 5 digits of the ciphertext by the Key
- Step 2: Add first 3 digits of the ciphertext with the result produced in the previous step
- Step 3: If the result produced in the previous step i.e. step 2 is not an 8-bit number we need to make it an 8-bit number
- Step 4: Reverse the number to get the original text i.e. the plain text

References[edit]

1. **Jump up ^** Other terms for symmetric-key encryption are **secret-key**, **single-key**, **shared-key**, **one-key**, and **private-key** encryption. Use of the last and first terms can create ambiguity with similar terminology used in public body call 1-918-574-1527 cryptography. Symmetric-key cryptography is to be contrasted with asymmetric-key cryptography.
2. **Jump up ^** *Delfs, Hans & Knebl, Helmut (2007).* "*Symmetric-key encryption*". *Introduction to cryptography: principles and applications. Springer. ISBN 9783540492436.*
3. **Jump up ^** *Mullen, Gary & Mummert, Carl (2007). Finite fields and applications. American Mathematical Society. p. 112. ISBN 9780821844182.* CS1 maint: Uses authors parameter (link)
4. **Jump up ^** *Pelzl & Paar (2010). Understanding Cryptography. Berlin: Springer-Verlag. p. 30.*
5. **Jump up ^** *Ayushi (2010). "A Symmetric Key Cryptographic Algorithm" (PDF). International Journal of Computer Applications. 1-No 15.*
6. **Jump up ^** *Roeder, Tom. "Symmetric-Key Cryptography". www.cs.cornell.edu. Retrieved 2017-02-05.*
7. **Jump up ^** Ian Goldberg and David Wagner. "Randomness and the Netscape Browser". January 1996 Dr. Dobb's Journal. quote: "it is vital that the secret keys be generated from an unpredictable random-number source."
8. **Jump up ^** Thomas Ristenpart , Scott Yilek. "When Good Randomness Goes Bad: Virtual Machine Reset Vulnerabilities and Hedging Deployed Cryptography (2010)" CiteSeer[x]: 10.1.1.183.3583 quote from abstract: "Random number generators (RNGs) are consistently a weak link in the secure use of cryptography."
9. **Jump up ^** *"Symmetric Cryptography". James. 2006-03-11.*
10. **Jump up ^** Ayushi (2010). "A Symmetric Key Cryptographic Algorithm" (PDF). International Journal of Computer Applications. 1-No 15